United States Military Academy
West Point, New York 10996

A HYBRID EXPERT SYSTEM FOR SCHEDULING THE US ARMY'S CLOSE COMBAT TACTICAL TRAINER (CCTT)

OPERATIONS RESEARCH CENTER
TECHNICAL REPORT 95-10-1

by

Lieutenant Colonel Michael L. McGinnis

Major Robert G. Phelan Jr

September 1996

The Operations Research Center is supported by the Assistant Secretary of the Army (Financial Management & Comptroller)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September 1996

3. REPORT TYPE AND DATES COVERED
Technical Report

4. TITLE AND SUBTITLE
A Hybrid Expert Systems For Scheduling the US Army Close Combat Tactical Trainer (CCTT)

5. FUNDING NUMBERS

6. AUTHOR(S)
LTC Michael McGinnis
MAJ Robert Phelan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USMA Operations Research Center
West Point, New York 10996

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
95-10-1

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A.
Approved for public Release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The search for new, innovative approaches to training has led the US Army Simulation, Training and Instrumentation Command (STRICOM) in Orlando, Florida to develop a new family of high technology, computer-based training facilities called the Combined Arms Tactical Trainer (CATT). These training facilities will support realistic, interactive, combined arms training involving army units at the same or different locations operating in a virtual environment on a digitized battlefield.

The first of the CATT family to be developed will be the Close Combat Tactical Trainer facility featuring manned simulator modules for training armored and mechanized forces at battalion and below. Two versions of the CCTT system are under development: fixed-site system and mobile systems. Figure 1 below depicts the general layout and major components of a fixed-site CCTT facility.

The manned simulator modules of the CCTT include main battle tanks, armored personnel carriers and command and control vehicles. The operations center emulates and controls battlefield operating systems that habitually train with the unit undergoing training. The after action review consoles replay segments of a training scenarios for the training unit to illustrate training lessons. The master control console starts and ends the training scenarios, and controls the tempo of the training scenario as well. The maintance console monitors the systems and conducts system diagnostics. There are also consoles for controlling the computer generated, and semiautomatic forces that populate the digitized battlefield during scenario execution.

14. SUBJECT TERMS
Close Combat Tactical Trainer

15. NUMBER OF PAGES
143

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

A HYBRID EXPERT SYSTEM FOR SCHEDULING THE US ARMY'S CLOSE COMBAT TACTICAL TRAINER (CCTT)

Lieutenant Colonel Michael L. McGinnis

Major Robert G. Phelan Jr.

A TECHNICAL REPORT
OF THE
OPERATIONS RESEARCH CENTER
UNITED STATES MILITARY ACADEMY

Directed by
Lieutenant Colonel Michael L. McGinnis, Ph.D.
Director, Operations Research Center

Approved by
Colonel James L. Kays, Ph.D.
Professor and Head
Department of Systems Engineering

The Operations Research Center is supported by the Assistant Secretary of the Army (Financial Management& Comptroller)
The Sponsor for this project is the U.S. Army Simulation, Training and Instrumentation Command (STRICOM).

ACKNOWLEDGMENTS

This research was supported by the Office of the Assistant Secretary of the Army for Financial Management and Comptroller. The authors are especially thankful for advice and contributions to this paper made by the following individuals: Colonel (ret.) Bob White of the Institute for Defense Analysis, Lieutenant Colonel Joe Hughes formerly the SIMNET Director at Fort Knox, Major George Stone of STRICOM, and 2d Lieutenant Heidi Trush formerly of the US Military Academy. Finally, thanks to Meghan McGinnis and Matthew McGinnis for their assistance with development of the scheduling equation given by Equation (14) of Appendix A, and the heuristic program for scheduling multiple training scenarios simultaneously within the Close Combat Tactical Trainer (CCTT).

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

With the end of the Cold War, the US Army shifted from a forward-deployed force that, for nearly fifty years, mostly trained for a conventional, highly mechanized war in central Europe to a more centrally-based force responsible for a substantially broader range of military operations potentially occurring in any theater with little or no warning. These new challenges, combined with force structure downsizing and defense budget cuts, forced the Army to research new, innovative approaches to training. In turn, these efforts led to a family of high technology, computer-based training facilities called the *Combined Arms Tactical Trainer* (CATT). The first of these to be developed is the *Close Combat Tactical Trainer* (CCTT) facility; a synthetic, computerized environment for training armored and mechanized forces at battalion and below. Within the CCTT, a distributed interactive simulation (DIS) network connects virtual and constructive simulations, and manned simulator modules. The major tasks for planning a days' training include selecting scenarios for training; then scheduling the scenarios throughout planning horizon where multiple scenarios may be scheduled simultaneously, and where the training scenario duration varies by scenario type; and, finally, scheduling the type and quantity of training resources for conducting each scenario where resource quantities vary by scenario type and may vary within a scenario type as well. CCTT training resources include manned simulator modules, computer workstations, workstation operators (people), and computer generated, semi-automated forces (SAF). This report discusses the development of a hybrid expert system for selecting training scenarios and scheduling

(1) training scenarios and (2) a single resource for each scenario, semi-automated forces (SAF), via heuristic procedures.

1. INTRODUCTION

For the United States Army, the end of the Cold War also ended fifty years of organizing and training a forward-deployed armored force to fight a conventional, highly mechanized war in central Europe. Since then, the Army has made the transition to a more centrally based force responsible for a substantially broader range of military operations. Today, the possibility of crisis situations erupting in any theater of the world with little or no warning challenges Army leaders as never before to maintain the readiness of our armed forces. In recent years, the United States Armed Forces have been confronted by a series of diverse military operations. These have ranged from conventional war during Desert Storm to operations other than war in Rwanda, Somalia, Haiti, Cuba, and Bosnia. The likelihood that this trend will continue into the 21st Century is reflected in the US Army's Training and Doctrine Command (TRADOC) Pamphlet 525-5, *Future Full-Dimensional Operations*. Additionally, we should expect further reductions to both military force structure and defense budgets to continue for some time to come. This formidable set of circumstances demands that military commanders at all levels maintain the operational readiness of their forces by effectively and efficiently managing increasingly scarce training resources.

1.1 CLOSE COMBAT TACTICAL TRAINER (CCTT)

The search for new, innovative approaches to training has led the US Army Simulation, Training, and Instrumentation Command (STRICOM) in Orlando, Florida to develop a new family of high technology, computer-based training facilities called the *Combined*

Arms Tactical Trainer (CATT). These training facilities will support realistic, interactive, combined arms training involving army units at the same or different locations operating in a virtual environment on a digitized battlefield.

The first of the CATT family to be developed will be the *Close Combat Tactical Trainer* (CCTT) facility featuring manned simulator modules for training armored and mechanized forces at battalion and below. Two versions of the CCTT system are under development: fixed-site system and mobile system. Figure 1 below depicts the general layout and major components of a fixed-site CCTT facility.

The manned simulator modules of the CCTT include main battle tanks, armored personnel carriers, and command and control vehicles. The operations center emulates and controls battlefield operating systems that habitually train with the unit undergoing training. The after action review consoles replay segments of a training scenario for the training unit to illustrate training lessons. The master control console starts and ends the training scenarios, and controls the tempo of the training scenario as well. The maintenance console monitors the system and conducts system diagnostics. There are also consoles for controlling the computer generated, and semi-automated forces (SAF) that populate the digitized battlefield during scenario execution. A local area network will connect each major system component. For further details of the system, see US Naval Training Systems Center, 1992.

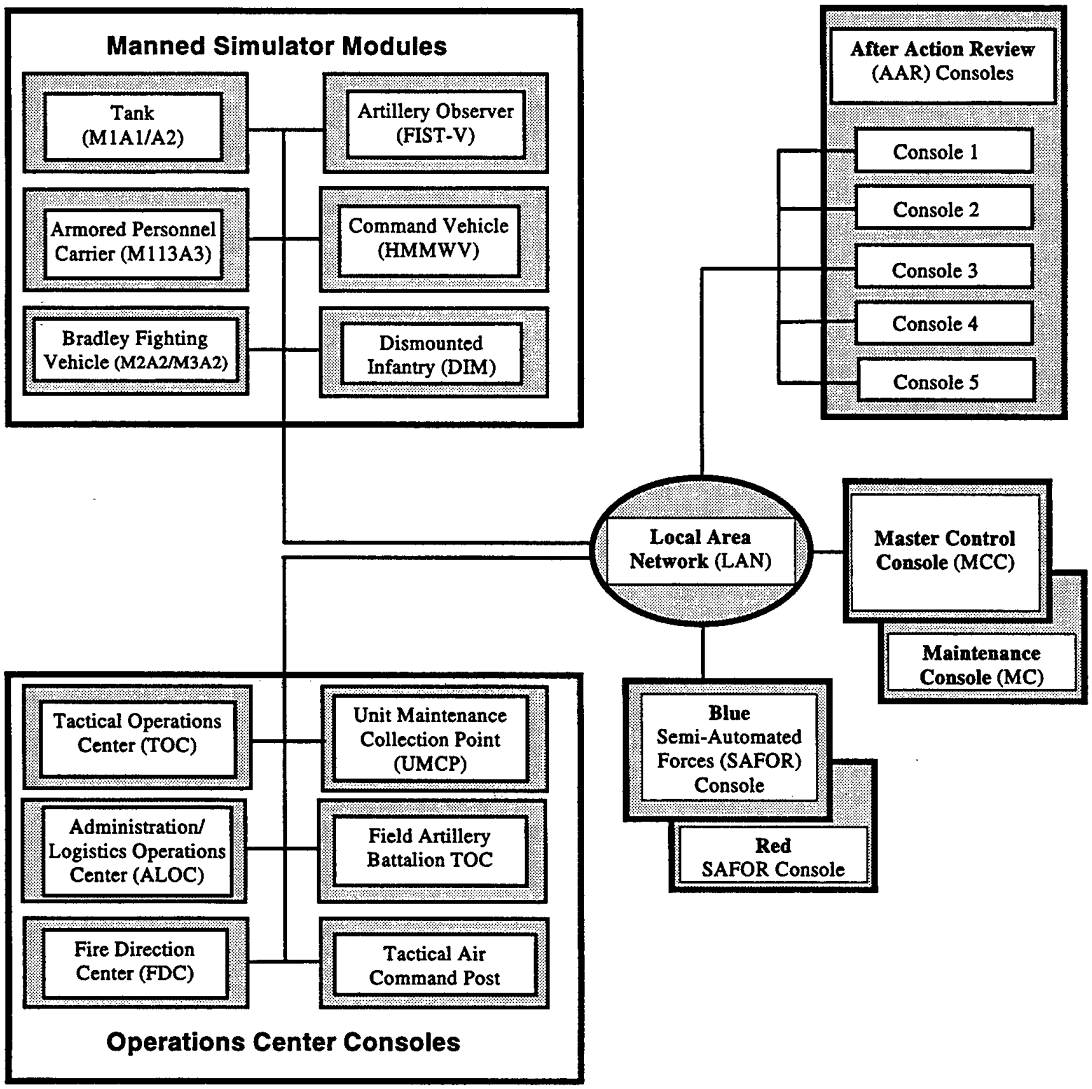

Figure 1. Fixed-site Close Combat Tactical Trainer layout and system diagram.

Eleven fixed-site facilities are scheduled for development. Eight will be located within the continental United States at Fort Rucker, Alabama, Fort Benning and Fort Stewart, Georgia, Fort Riley, Kansas, Fort Campbell, Kentucky, Fort Bliss and Fort Hood, Texas, and Fort Lewis, Washington. The three overseas sites include two in Germany and one in South Korea. Eventually, a distributed interactive simulation network will integrate all CCTT sites (Hollis & Shoffner, 1992).

1.2 TRAINING EXERCISE DEVELOPMENT SYSTEM (TREDS)

Another STRICOM program related to the Close Combat Tactical Trainer system is the *Training Exercise Development System* (Crissey, Stone, Briggs, & Mollaghasemi, 1994). TREDS is an intuitive, user-friendly, computer software system for planning and executing CCTT training. When fully developed, TREDS will automate many tasks for planning training previously done by hand. Figure 2 below outlines the major phases of planning and executing CCTT training with TREDS, and the key steps within each phase (McGinnis & Stone, 1995).

In Phase I, the training unit appraises its training proficiency as a preliminary step to scheduling CCTT training. The training unit produces a *training task list* that identifies and prioritizes essential unit training tasks. In Phase II, the unit uses the training task list from Phase I to generate a prioritized, unconstrained list of training events called the *training event list*. In Phase III, the training unit selects and schedules training scenarios that reflect the training events of Phase II. The training unit also selects and schedules training resources for executing the training scenarios. During resource scheduling, a training resource shortfall occurs whenever the level of resources required to conduct training exceeds resources available. Currently within TREDS, training unit personnel use manual, trial-and-error methods to resolve training resource shortfalls.

Phase I. Select Tasks for Training

<u>Step 1</u>. Assess unit training proficiency.
<u>Step 2</u>. Identify essential (major) training tasks.
<u>Step 3</u>. Identify key (lesser) training tasks.
<u>Step 4</u>. Develop a prioitized training tasks list.

Phase II. Build a Preliminary List of Training Events

<u>Step 1</u>. Given Step 4 of Phase I, identify a preliminary set of training events.
<u>Step 2</u>. Develop a preliminary prioritized training events list.

Phase III. Build the Training Scenario

<u>Step 1</u>. Given the prioritized training task and event lists, select the "best" training scenario from TREDS' library of training scenarios.
<u>Step 2</u>. Modify the training scenario selected above (if necessary) by incorporating training tasks from Phase I not currently included in the training scenario.
<u>Step 3</u>. From the modified training scenario of Step 2, identify training resources required to conduct training.
<u>Step 4</u>. Match the available training resources against the current requirement for training resources.

<u>IF</u>: resources available ≥ resources required;

<u>THEN</u>: done. Schedule the training scenario.

<u>IF</u>: available resources < required resources;

<u>THEN</u>:

<u>EITHER</u>: modify the training scenario under consideration to make resources available ≥ resources required;

<u>OR</u>: select another scenario and return to Step 2.

Phase IV. Execute the Training Scenario and Measure Training Performance

Phase V. Conduct Training After Action Review (AAR)

Figure 2. TREDS Training Planning System Model.

For example, the system user may resolve a training resource shortfall by selectively overriding training resource levels recommended by TREDS. Alternatively, the user may also select a new training scenario that requires fewer training resources to execute. Finally, the user may reduce resource requirements by simply deleting a

scenario from the current schedule to make more resources available. Unfortunately, the complex nature of the CCTT scheduling problem makes resolving training resource shortfalls by hand a very formidable task (see Section 2). This motivated the development of automated methods for scheduling resources and resolving scheduling conflicts within TREDS.

During training scenario execution in Phase IV, TREDS will automatically collect data on training performance measures established in advance by training unit personnel. In Phase V, TREDS analyzes training data collected during Phase IV for the after action review (AAR). The AAR results may then be used in the training assessment phase of the next iteration of the CCTT training cycle.

This paper addresses the problem of scheduling training scenarios and a single training resource for fixed-site systems. However, similarities between fixed and mobile sites suggest that the scheduling methods may apply to mobile sites as well. Section 2 covers the dynamics of the scheduling problem, related work, and major assumptions and constraints. Section 3 formulates a mathematical model of the scheduling problem, and presents several scheduling objectives. Section 4 discusses the development of a hybrid expert system prototype, integrated within TREDS, for scheduling scenarios and training resources. Section 5 suggests areas of future work for improving the hybrid expert system developed here for scheduling CCTT training.

2. THE CCTT SCHEDULING PROBLEM

Experts estimate that once all CCTT facilities are operational, Army units at battalion and below will have, on average, twenty days of CCTT training per year. Effective scheduling of the facilities is critical to training success. This is especially important to Active and National Guard units using the facilities to sustain unit readiness given future reductions to defense operations and maintenance funds. Proper CCTT facility utilization, that simultaneously maximizes training opportunity, will require quick and efficient methods for scheduling and rescheduling training.

CCTT scheduling involves three interdependent decision elements:

1. selecting training scenarios to schedule (*training scenario selection problem*);
2. scheduling training scenarios throughout the training day (*training scenario scheduling problem*); and
3. selecting and scheduling training resources for each scenario (*training resource scheduling problem*).

Given these tasks, it may appear that making scheduling decisions ought to be a fairly straightforward process. However, significant obstacles exist to making "good" scheduling decisions in a dynamic environment where training continues throughout the day. For example, unforeseen breakdowns and training delays may disrupt training scheduled and invalidate the current training schedule. This, in turn, may require rescheduling training scenarios and training resources previously scheduled for future periods. The temporal interdependence of scheduling decisions also complicates scheduling where future scheduling decisions depend upon decisions made in the current period.

2.1 RELATED WORK

A survey of the literature on military training and scheduling failed to produce papers directly related to the CCTT scheduling problem. However, several studies suggested approaches for modeling the problem. For example, Yang and Ignizio (1987) solved a somewhat related problem that involved scheduling training resources and training activities for a fixed number of US Army battalions located at the same training installation. Their problem featured constraints on the type and quantity of training resources to be scheduled. Precedence relationships also existed between tasks and training units. These occasionally required two or more battalions to work together and share resources in accomplishing certain training tasks. The authors scheduled training activities and resources using a heuristic program applied in two phases.

McGinnis and Fernandez (October 1994, December 1994) solved a scheduling problem for the US Army's Basic Combat Training phase of initial entry training. Their problem dealt with simultaneously scheduling hundreds of training companies at different training installations across the United States. The scheduling problem was to determine how many recruits to assign to each training company in each week of a two-year planning horizon. The authors formulated a dynamic system model of Basic Combat Training for optimally scheduling training resources based on dynamic programming. In addition, McGinnis and Fernandez developed a three-phase heuristic procedure that they implemented in a decision support system for efficiently solving real-world scheduling problems. Performance measures include training quality, resource utilization, and training cost.

Boldovici and Bessemer (1993) discussed research related to the development of a prototype CCTT facility at Fort Knox, Kentucky, called the *Simulation Networking* (SIMNET) facility. The author's summarized SIMNET research on training effectiveness including lessons learned for designing future computer-based training facilities and developing realistic training scenarios. Boldovici and Bessemer's discussion of SIMNET training effectiveness helped identify appropriate scheduling objectives for the CCTT scheduling problem. Finally, we refer the reader to Brown and Scherer (1995) for a discussion of intelligent scheduling systems.

2.2 SCHEDULING DYNAMICS

2.2.1 Varying Scheduling Horizon.

According to CCTT requirements documentation, weekday and weekend training last nine hours (0800 to 1700) and ten hours (0800 to 1800), respectively (US Army STRICOM, 1991). However, CCTT developers suggest the duration of a training day may vary from 6 to 24 hours depending upon the needs of the training unit and availability of the facility. For expert system prototype development, we simplified the problem by assuming a bounded training day, up to 24 hours in duration, with no breaks, where training scenarios start and stop instantaneously on the hour. This scheduling horizon was determined to be adequate for evaluating the functionality of the expert system prototype in advance of full system development.

2.2.2 *Training Scenario Duration.*

Planning a days' training involves scheduling up to five simultaneous training scenarios that vary in duration according to three types of tactical exercises. Table 1 gives the duration of each tactical exercise by training unit echelon.[1]

Table 1. Training scenario duration.

Echelon/Exercise	Movement to Contact	Deliberate Attack	Defend in Sector
Platoon	2 hours	2 hours	2 hours
Company	3 hours	3 hours	3 hours
Battalion	3 hours	5 hours	6 hours

2.2.3 *Type of Training.*

Observations of SIMNET training and discussions of training issues with system managers suggested two approaches for scheduling CCTT training: *fixed and sequential training* (FAST), and *flexible iterative training* (FIT). During fixed and sequential training (FAST), a partially or fully trained unit executes a fixed sequence of training scenarios selected and scheduled ahead of time for the entire planning horizon. A FAST schedule specifies the number and type of training scenarios to start in each hour t of the training day and the type and quantity of training resources by scenario.

In flexible iterative training (FIT), the training unit selects and schedules an initial set of training scenarios based on the unit's training proficiency assessment during Phase I of the CCTT training cycle (see Figure 2). Future scenario and resource scheduling

[1] The five-hour duration of a battalion deliberate attack exercise (see Table 1) is actually 4.5 hours. However, using a five hour scenario is consistent with the assumption made in Section 2.3 that all scenarios start at the beginning of a training period; i.e., on the hour.

decisions are made dynamically based on several factors. Most important are performance results from CCTT training and feedback provided during the after action review. Other considerations include the level of training resources available at the time a scheduling decision is made and the time left in the planning horizon for scheduling training scenarios.

2.2.4 Training Scenario Categories.

- Training scenario categories are listed below for selecting scenarios during a consultation with the expert system. The categories reflect unit training information used within the expert system prototype to categorize scenarios and criteria for selecting them.

- *Unit type* (4) specifies the type of unit undergoing training: armor, mechanized infantry (MECH), light infantry (LT INF), cavalry troop (CAV).

- *Unit echelon* (3) denotes the level of the training unit: platoon (PLT), company (CO), battalion (BN).

- *Training context* (5) identifies other friendly (BLUE) units participating in the training scenario with the unit undergoing training. However, these "other" BLUE forces populate the digital battlefield as computer generated forces and are controlled by a BLUE workstation operator. The training contexts include: platoon within platoon (PWP), platoon within company (PWC), company within company (CWC), company within battalion (CWB), and battalion within battalion (BWB).

  The training context of a unit within itself, such as platoon within platoon, implies the unit undergoing training populates the digitized battlefield by itself. When training involves a unit within a higher echelon unit, such as platoon within company, then the training unit will "see" other computer generated units on the digitized

battlefield. For example, a platoon within company, the training unit would see the other two computer-generated platoons from the training unit's company.

- *Exercise type* (3) characterizes scenarios according to the major tactical event conducted during a training scenario: movement to contact (MTC), deliberate attack (A), defend in sector (D).

- *Training proficiency* (3) refers to the training unit's overall level of training by exercise type: trained (T), partially trained (P), untrained (U).

- *Opposing force echelon* (5) identifies the allowable computerized RED forces that may oppose a BLUE force: squad (SQD), platoon (PLT), company (CO), battalion (BN), regiment (REGT).

Table 2 enumerates the combinations of training unit echelon, training context and red force echelon for selecting training scenarios (reading left to right). Note only three of five red force echelons are allowed. All together, these generate 1620 possible training scenario categories $(4x3x5x3x3x3)$. In the future, there may be multiple scenarios per category. Once fully developed, the expert system will enable the system user to methodically identify good training scenarios tailored to the unique needs of the training unit.

Table 2. Training unit, training context, and red force echelon combinations.

Training Unit Echelon	Training Context	Opposing Force Echelon 1	Opposing Force Echelon 2	Opposing Force Echelon 3
platoon	platoon	squad	platoon	company
platoon	company	platoon	company	battalion
company	company	platoon	company	battalion
company	battalion	company	battalion	regiment
battalion	battalion	company	battalion	regiment

2.2.5 Varying Demand for Training Resources.

For the expert system prototype, the resource scheduling problem has been restricted to scheduling semi-automated forces (SAF) hour-by-hour throughout a one day planning horizon. The demand for SAF in any hour t depends upon two factors. One is the number and type of training scenarios to start at the beginning of hour t. Second, the number and type of training scenarios from earlier periods that are still busy at the beginning of hour t. In any period, up to five training scenarios may be conducted simultaneously. The decision sequence for solving this problem requires that training scenario selection and scheduling decisions be made prior to the training resource scheduling decisions. This sequence makes the training resource scheduling problem completely deterministic.

2.2.6 Dynamics of Varying Computer Generated, Semi-automated Forces (SAF).

Another important aspect of scheduling is determining SAF strength, the number of friendly (BLUE) and opposing (RED) semi-automated forces (SAF) to assign to each training scenario scheduled throughout the planning horizon. BLUE SAF strength varies by training scenario category (see above) making it an important decision element of the training scenario scheduling problem. CCTT experts have established fixed BLUE SAF strengths for each training scenario category based on training realism and CCTT system constraints (see Table 3 below). Therefore, BLUE SAF strength is not a decision element of the resource scheduling problem. However, RED SAF strength is a decision element of both the scenario scheduling and resource scheduling problems.

RED SAF strength varies by training scenario category, and varies between upper and lower bounds within a training scenario category as well. Here, we bound RED SAF strength above by a full-strength RED force that a trained BLUE force would expect to face in a real-world military operation. The RED SAF lower bound reflects what experts consider to be a reasonably sized RED force to oppose an untrained blue force. For expert system development, CCTT experts set RED SAF lower bounds at 60% of a full-strength RED force. The upper and lower bounds for RED SAF vary by training scenario category. Table 3 gives RED SAF upper bound values by training category and the requisite number of BLUE SAF by training scenario category for two types of units: armor and mechanized infantry (MECH). The bold entries of Table 3 highlight the cases where differences exist between *ARMOR* and *MECH* SAF strengths for the same training scenario category. The right most column, *TOTAL SAF*, represents the maximum number of BLUE and RED SAF (combined) that may be assigned to a training scenario for a given category.

Table 3. SAF entities by training unit, training context and red force echelon.

Training	Training	Red Force	Red SAF		Blue SAF		Total SAF	
Unit	Context	Echelon	Armor	Mech	Armor	Mech	Armor	Mech
platoon	platoon	squad	23	23	0	0	23	23
		platoon	20	20	0	0	20	20
		company	65	65	0	0	65	65
platoon	company	platoon	29	29	39	39	68	68
		company	62	62	12	12	74	74
		battalion	176	176	**23**	**27**	**199**	**203**
company	company	platoon	29	29	27	27	56	56
		company	62	62	0	0	62	62
		battalion	176	176	**11**	**15**	**187**	**191**
company	battalion	company	65	65	**64**	**88**	**121**	**153**
		battalion	138	138	**73**	**93**	**211**	**231**
		regiment	491	491	**85**	**101**	**129**	**592**
battalion	battalion	company	65	65	0	0	65	65
		battalion	138	138	9	9	147	147
		regiment	491	491	17	17	508	508

Determining RED SAF strength for a training scenario that starts in hour t requires the following information:

- the number and type of simultaneous training scenarios to start in hour t;
- the number of RED SAF available at the beginning of hour t; and
- the priority for scheduling RED SAF by training scenario in each hour t.

Demand for RED SAF is measured by the number of RED SAF needed to execute the training scenarios that start in each hour. A shortfall in RED SAF, or other resource, arises when the demand for RED SAF (at the beginning of an hour) exceeds the level of resources available. Therefore, the number of RED SAF available for scheduling at the beginning of hour t obviously depends upon previous RED SAF strength decisions. This information must be accounted for when making scheduling decisions which complicates the decision process.

For example, RED SAF scheduling decisions in previous periods that fixed RED SAF strengths at or near the upper bound values (see Table 3) leave fewer RED SAF available for scheduling in hour t. As just mentioned, this may cause a shortfall in RED SAF in any period of the planning horizon, except the first period. RED SAF scheduling shortfalls are resolved by sequentially revising RED SAF strengths from previous periods (if possible) to make more RED SAF available to meet demand in the current period. In any period, if no further adjustments to RED SAF strengths are possible, then the training resource shortfall cannot be corrected and the training resource schedule is *infeasible*.

Currently, two options exist for correcting a RED SAF shortfall. First, the system user may replace the training scenario being scheduled in the current period, or one from a previous period, with a scenario that requires fewer RED SAF. Second, the system user may delete a scenario from the schedule to make more RED SAF available in the current period.

2.2.7 Scheduling Constraints.

Scheduling involves two types of constraints. The first is a *user preference constraint* that prioritizes the user's preferences for scheduling training scenarios throughout the planning horizon. This reflects the prioritized training task and event lists from Phases I and II of the TREDS training planning system model shown in Figure 2. The second, a *training exercise precedence constraint*, specifies ordered sequences for conducting training exercises that, in turn, establish rules for scheduling training scenarios. For example, the system user may establish precedence constraints for certain training scenario contexts such as movement to contact must precede an attack. Constraints such as these have been incorporated into the expert system knowledge base as rules for scheduling training scenarios (see Section 3).

2.3 MAJOR SCHEDULING ASSUMPTIONS AND CONSTRAINTS

2.3.1 Modeling Assumptions

- Completely deterministic scheduling problem.
- Finite scheduling horizon of T consecutive, discrete, one-hour periods.
- Scenario duration varies according to the duration of exercises given in Table 2.

- Training scenarios always start at the beginning of hour *t*.
- Fixed number of BLUE SAF for each scenario is fixed by training category.
- Varying, but bounded, number of RED SAF for each scenario by category.
- No backlogging of demand for RED SAF.

2.3.2 Modeling Constraints

- Up to five training scenarios may be conducted simultaneously in any hour *t* of the scheduling horizon. Scheduling simultaneous scenarios is subject to the following constraints: (1) a battalion scenario is exclusive of all other scenarios; (2) up to three company scenarios may be scheduled during the same period; and (3) when scheduling five simultaneous scenarios in a period, if three are company level, then the remaining two must be platoon level scenarios.

- The number of semi-automated force (SAF) entities controlled by a single operations center (OC) workstation (console) in any hour *t* is bounded above at 303.

- The number of semi-automated force (SAF) entities scheduled in any hour *t* is bounded above at 600.

- The number of semi-automated force (SAF) entities controlled by a human controller at a console in any hour *t* is bounded above at 60.

- The number of manned simulator modules scheduled for use in any hour *t* is bounded above at 37.

3. MATHEMATICAL MODEL

This section presents a mathematical formulation of a simplified CCTT scheduling problem. The model is limited to a single fixed scheduling horizon, and a single training resource (i.e., RED SAF).

3.1 MATHEMATICAL NOTATION

T: Number of hours in the scheduling horizon (*training day*);

t: Period (*hour*) of the scheduling horizon, $t \in \{1, 2, \ldots, T\}$;

J: Total number of training scenario categories, where J represents the enumeration of all combinations of $\{k, l, m, n, o, p\}$ (see below);

j: Index for training scenario categories, $j \in \{1, 2, \ldots, J\}$. Each j represents a training scenario category generated by enumeration of training scenario sub-categories (see Section 2.2).

$d_j(t)$: Duration, in hours, of training scenario $\tilde{s}$ (see below), type j, scheduled in hour t.

D: Set of allowable time intervals, in hours, specifying the duration of a scenario, where $D \equiv \{2, 3, 5, 6\}$. The duration of a specific scenario is determined by its type j.

$\tilde{D}$: Set of allowable time intervals for possible combinations of training scenarios of type j in any time period (see Table 1); $\tilde{D} \equiv \{(2), (2,3), (5), (6)\}$.

K: Total number of types of training units;

k: Index for training unit type, $k \in \{1, 2, \ldots, K\}$;

L: Total number of training unit echelons;

l: Index for training unit echelon, $l \in \{1, 2, \ldots, L\}$;

M: Total number of training contexts;

m: Index for training context, $m \in \{1, 2, \ldots, M\}$;

O: Total number of opposing (RED) force echelons;

o: Index for training opposing force echelon, $o \in \{1,2,\ldots,O\}$;

P: Total levels of training unit proficiency;

p: Index for training unit proficiency, $p \in \{1,2,\ldots,P\}$;

R: Total number of RED SAF available within the CCTT system;

$\overline{R}(t)$: Total allowable number of RED SAF that may be busy in any hour t;

$\overline{r}_j(t)$: Upper bound for RED SAF scheduled in hour t for scenario j;

$r_j(t)$: Number of RED SAF to start a training scenario in hour t for scenario j;

$\underline{r_j}(t)$: Lower bound for RED SAF scheduled to start in hour t for scenario j;

I: Total number of possible idle RED SAF;

$I(t)$: Number of RED SAF idle at the beginning of hour t;

$s(t)$: Number of training scenarios scheduled to start in hour t;

$\tilde{s}_j(t)$: Specific training scenario of type j scheduled in hour t; and

$\overline{S}(t)$: Upper bound for the number of simultaneous training scenarios scheduled, by echelon l, in any hour t.

3.2 MATHEMATICAL FORMULATION OF THE PROBLEM

3.2.1 Stages.

In the CCTT scheduling problem, stages (periods) are specified by hour t of the training day. The planning horizon T consists of a finite number of identical, discrete time periods where $t \in \{1,2,\ldots,T\}$.

3.2.2 Modeling Constraints.

$$\underline{r_j}(t) \le r_j(t) \le \overline{r}_j(t) \quad \forall \quad (j,t): \tag{1}$$

RED SAF scheduling constraint by training scenario type j;

$$0 \le \sum_{j=1}^{J} r_j(t) \le \overline{R}_j(t) \quad \forall \quad (j,t): \tag{2}$$

RED SAF constraint on the allowable number of RED SAF that may be busy during hour t;

$I \leq R$: conservation of RED SAF constraint; (3)

$$0 \leq \sum_{l=1}^{L} s_l(t) \leq \overline{S}_l(t) \quad \forall \quad (l,t): \tag{4}$$

scenario scheduling feasibility constraint determined by training unit echelon l;

$0 \leq I(t) \leq I \quad \forall \quad (t)$: scheduling feasibility constraint. (5)

The number of training scenarios "busy" in any period that started in the "current" period $\hat{t}$, or an earlier period $\hat{t} - d_j(t)$, is given by

$$S(\hat{t}) = s_j(\hat{t}) + \sum_{d_j(t) \in D} s_j(\hat{t} - d_j(\hat{t})): \tag{6}$$

The number of RED SAF that become available in hour t having just completed a training scenario starting $d_j(t) \in D$ hours earlier is

$$\sum_{d_j(t) \in D} r(t - d_j(t)) \quad \forall \quad t > d_j(t). \tag{7}$$

For example, for the "current" period $\hat{t}$, the values of $\hat{t} - d_j(t)$ reflect the number of hours earlier in the scheduling horizon when a group of RED SAF started a training scenario in hour. The RED SAF remain busy for $d_j(t)$ hours until the training scenario ends in hour t. At the beginning of hour $t+1$, the RED SAF become idle and are again available for scheduling. The training scenario times given in Table 1 and system scheduling constraints of Section 2.3 establish the allowable combinations of battalion, company, and platoon scenarios. The possible values for $d_j(t)$ in any hour t are

specified by set $\tilde{D} \equiv \{ (2),(2,3),(3),(5),(6),(\phi) \}$. The set $\tilde{D}$ enumerates the *allowable* time intervals, and coinciding combinations of scenarios, that may occur in any time period t of the planning horizon.

$\tilde{D} = (2)$: Indicates up to five groups of RED SAF become available in hour $t+1$; those groups that just finished 2-hour, platoon-level scenarios only.

$\tilde{D} = (3)$: One group of RED SAF becomes available in hour $t+1$; the one that just completed a 3-hour, battalion, movement to contact scenario.

$\tilde{D} = (2,3)$: Up to five groups of RED SAF become available in hour $t+1$; the groups that just finished either 3-hour company or 3-hour battalion, or 2-hour platoon scenarios, or a combination of the two. Possible combinations of company (CO) and platoon (PLT) scenarios lasting three and two hours, respectively, that would finish in hour $t+1$ are given in Table 4.

Table 4. Training unit combinations for simultaneous scenarios.

5 Scenarios	4 Scenarios	3 Scenarios	2 Scenarios	1 Scenario
3 co-2 plt	3 co-1 plt	3 co	2 co	1 co
2 co-3 plt	2 co-2 plt	2 co-1 plt	1 co-1 plt	1 plt
1 co-4 plt	1 co-3 plt	1 co-2 plt	2 plt	
5 plt	4 plt	3 plt		

$\tilde{D} = (5)$: One group of RED SAF becomes available in hour $t+1$; the one that just completed a 5-hour, battalion, deliberate attack scenario.

$\tilde{D} = (6)$: One group of RED SAF becomes available in hour $t+1$; the one that just completed a 6-hour, battalion, defend in sector scenario.

$\tilde{D} = (\phi)$: Indicates that for values of $t \le d_j(t)$, $\sum_{d_j(t) \in D} r(t - d_j(t)) = 0$.

3.2.3 State Transition Equation.

The state of the training system evolves according to the following balance equation for idle training resources; RED SAF in this case:

$$I(t+1) = I(t) + \sum_{j=1}^{J} \sum_{d_j(t) \in D} r_j(t - d_j(t)) - \sum_{j=1}^{J} r_j(t) \quad \forall \quad t > d_j(t). \qquad (8)$$

$r_j(t)$ is the number of RED SAF scheduled for scenario j starting in hour t.

$\sum_{d_j(t) \in D} r_j(t - d_j(t))$ for $t > d_j(t)$ represents the number of RED SAF scheduled for training scenario(s) of type j available at the beginning of hour $t+1$ to start another training scenario having just completed one that began either two, three, five, or six hours earlier (see Table 2 and Equation 6 above). In all other cases for $t \le d_j(t)$, $\sum_{d_j(t) \in D} r(t - d_j(t)) = 0$ and $\tilde{D} = \{\phi\}$.

3.2.4 Scheduling Decisions and Scheduling Policy.

Training scenario and training resource decisions, $s_j(t)$ and $r_j(t)$, respectively, are made at the beginning of hour t for $t = 1, 2, \ldots, T-1$. A sequence of such decisions, denoted by π, is represented by

$$\pi = \begin{Bmatrix} \pi_S \\ \pi_R \end{Bmatrix} = \begin{Bmatrix} s_j(1), \ldots, & s_j(t), \ldots, & s_J(T-1); \\ r_j(1), \ldots, & r_j(t), \ldots, & r_J(T-1) \end{Bmatrix}. \qquad (9)$$

The set of all such feasible sequences, denoted by Π, are those satisfying conditions (1) through (6) above. π_S and π_R represent the decision sequences for training scenarios and training resources for the planning horizon, respectively.

3.2.5 Objective Function.

The heuristic scheduling method (see Appendix A) iteratively improves the initial (or current) scheduling policy for training resources π_R, period by period, until further improvements are not possible. Although the heuristics do not guarantee convergence to an optimal scheduling sequence, they are designed to generate good suboptimal schedules. The "best" suboptimal sequence of scenario and resource scheduling decisions, denoted π_R^*, is the one (perhaps among others) that suboptimizes equations (11) and (12) below for a fixed initial state $I(0)$. It is denoted by Φ_π where

$$\Phi_{\pi_R^*}[I(0)] = \underset{\pi \in \Pi}{(sub)optimal}\ \Phi\pi_R[I(0)]. \qquad (10)$$

3.3 SCHEDULING OBJECTIVES

An important aspect of scheduling is choosing a scheduling objective. Ideally, one that benefits both CCTT managers and users. The appropriateness of a scheduling objective depends, in large part, upon the goals of who is doing the scheduling. For the CCTT, scheduling objectives important to managers may not be of interest to the unit undergoing training. For example, system managers may consider system utilization an important

scheduling objective, whereas, system users may be more interested in training schedules that maximize the quality of training.

The unique structure of the CCTT scheduling problem complicates the choice of a suitable objective function for measuring scheduling performance. As discussed in Section 2, we decomposed the scheduling problem into three interdependent subproblems. This makes it virtually impossible to devise a single scheduling objective that reasonably accounts for practical aspects of each subproblem. Therefore, separate objectives are formulated for each subproblem. An alternate objective that minimizes training support costs is also presented.

3.3.1 Training Scenario Selection Objective.

In future CCTT training, selecting good training scenarios will largely depend upon the system user's CCTT experience, training expertise, and subsequent knowledge of unit training proficiency. Interviews with subject matter experts revealed that selecting training scenarios is a highly subjective process. It depending greatly upon the experience of training unit personnel at using the CCTT and knowledge of unit training proficiency. Under these circumstances, we doubted that exact or suboptimal methods could be formulated for selecting scenarios in a way that would be meaningful to the training unit. Therefore, a utility function approach for selecting training scenarios that would accurately quantify the value of scenarios to system users in precise mathematical terms was not attempted (see Clemen, 1990). Instead, scenario selection was left to the system user, however, many tasks for selecting scenarios were automated during development of the expert system prototype.

3.3.2 Training Scenario Scheduling Objective.

The objective suggested here for scheduling training scenarios is one that maximizes training opportunity. Training opportunity is subjectively determined by measuring the "value" of a given scenario scheduling policy $\{ s_j(1),\ldots,s_j(t),\ldots,s_j(T-1) \}$ to the system user. This accounts for practical aspects of scheduling and meets two other criteria important to expert system development, as well. First, it appeals, at least to some extent, to both CCTT managers and training units. Second, it can be used to discriminate among competing feasible training scenario schedules.

The straightforward objective assumes a user-determined value $c_j(t)$ reflecting the contribution of each scenario to the unit undergoing training. Given this, the cumulative value Φ_{π_S} of for each sequence of scheduling decisions $\pi \in \Pi$ by category j and hour t gives a measure of training opportunity, where

$$\Phi_{\pi_S} = \sum_{t=1}^{T-1} \sum_{j=1}^{J} c_j(t). \tag{11}$$

The superscript S of Φ_{π_S} identifies equation (11) as the scenario scheduling objective value.

3.3.3 Training Resources Scheduling Objective.

The training resource scheduling objective used by the automated scheduling heuristics to generate acceptable resource schedules reflects a measure of training quality from the

training unit's viewpoint. Training quality is measured by RED SAF strength, i.e., the number of RED SAF assigned to each training scenario scheduled throughout the planning horizon. As discussed in Section 2.2, the expert system determines RED SAF strength from expert system knowledge and training unit information such the military unit's training proficiency assessment (T, P, U) by exercise type (see above).

For example, the number of RED SAF opposing a fully trained BLUE force would generally outnumber the RED SAF facing a partially trained force. In turn, the RED SAF opposing a partially trained BLUE force would exceed the number facing an untrained BLUE force. From this it follows that RED SAF strength can be used to measure the "quality" of competing training resource schedules for a training unit and serve as a means of comparing training schedules between training units as well. An equivalent measure of maximizing RED SAF strength used here is minimizing idle RED SAF for a fixed initial condition $I(0)$ as given by

$$\Phi_{\pi_R}\left[I(0)\right]=\sum_{t=1}^{T} I(t), \tag{12}$$

The superscript R of Φ_{π_R}, denotes the <u>R</u>esource scheduling objective value of equation (12) where equation (8) determines the number of idle RED SAF in each period.

3.3.4 Training Cost Objective.

Another important scheduling objective is minimizing training costs. Unfortunately, since none of the CCTT facilities are yet operational, real-world data was not available

for formulating a cost functional. Nevertheless, the Director for Logistics at STRICOM provided the study team with useful training support cost estimates we used in formulating a preliminary cost objective (see Table 5).[2] Training support costs, also referred to as contractor logistics support costs, include operator and maintenance support for all CCTT system components except for system operators for the Operations Center consoles and manned simulator modules (see Figure 1), and CCTT building maintenance costs. The manned modules will be operated by army personnel and the host installation will assume responsibility for CCTT building maintenance and utility costs.

Table 5 summarizes STRICOM's support cost estimates for fiscal years (FY) 1999 through 2003. The average training support costs per hour and per 12-hour training day computed from STRICOM's support cost estimates are also given.

Table 5. Fixed-site training support cost estimates.

Cost Category / Fiscal Year	1999	2000	2001	2002	2003
Estimated Monthly Support Cost	$46,353	$55,215	$59,370	$60,698	$61,978
Estimated Annual Support Cost	$556,236	$662,580	$712,440	$728,376	$743,736
Average Training Cost per Hour	$201	$239	$257	$263	$268
Average Cost per 12-Hour Day	$2,410	$2,870	$3,086	$3,156	$3,222

For simplicity, only four types of training costs are considered in formulating the scheduling objective for minimizing the cost of CCTT training. Cost functional notation is as follows

C^{FS}: Fixed "Setup" cost assessed at the beginning of the training day and at the start of each training scenario;

C^{VIR}: variable cost per Idle training resource;

[2]Courtesy of Mr. Corbett Myers, Logistics Program Manager, Directorate for Logistics, US Army STRICOM.

C^{VBR}: Variable cost per Busy training Resource;

C^{VC}: Variable cost per training scenario Category; and

$P_j(t)$: indicator variable, where $P_j(t) \in \{0,1\}$. $P_j(t) = 0$ when no training scenarios of type j begin in hour t, and one otherwise.

The problem is to determine the number of training scenarios to start each hour and RED SAF strength for each scenario that minimize total training cost. The cost objective function value Φ_{π_C}, denoted by superscript C, for a fixed initial condition $I(0)$ is given by

$$\Phi_{\pi_C}[I(0)] = C^{VIR}I(T) + C^{VBR}r_j(T) + C^{VC}s_j(T) + \sum_{t=0}^{T-1}\left[\ C^{FS}P_j(t) + C^{VIR}I(t) + C^{VBR}r_j(t) + C^{VC}s_j(t)\ \right]. \tag{13}$$

3.4 COMPLEXITY OF THE SCHEDULING PROBLEM

The scheduling dynamics discussed above cause a combinatorial explosion in the number of options for selecting and scheduling training scenarios. For example, the fully developed system may present system users with up to 1620 possible training scenario categories for selecting training scenarios (see Section 3.2). The possibility of several scenarios per category would most certainly overwhelm system users with choices for selecting scenarios.

Scheduling training scenarios is even more challenging. For example, the number of training scenario scheduling combinations for a single 12-hour planning horizon may be estimated by enumeration of the decision spaces given below.

- 128 $(4x4x4x3)$ combinations of multiple training scenarios per hour (see Table 4), where $2 \le s_j(t) \le 5 \ \forall \ (t)$.
- 5 options for scheduling a single scenario per hour.
- 11 options for starting a 2-hour scenario.
- 10 options for starting a 3-hour scenario.
- 7 options for starting a 5-hour scenario.
- 6 options for starting a 6-hour scenario.

Together, these generate $2.95x10^6$ possible scheduling combinations for the training scenario scheduling problem for a twelve-hour planning horizon. The problem becomes even more complex when considering all possible scheduling horizons in a 24-hour training day.

RED SAF scheduling suffers from similar problems. An upper bound estimate of possible RED SAF scheduling decision sequences enumerated for a twelve-hour scheduling horizon from the following: four training contexts, three red force echelons, two of four possible types of training units (ARMOR and MECH), and, fifty-six allowable RED SAF strengths, on average, per training context, red force echelon and training unit triplet. Together these generate approximately $3.47x10^{37}$ decision outcomes ($(4x3x2x56)^{12}$). Once again, the number of scheduling options increases significantly if we consider scheduling sequences for each of 19 additional scheduling horizons between 6 and 24 hours.

Various exact methods were initially considered as a means of solving the CCTT scheduling problem. Unfortunately, the size of the problem made these methods impractical for prototype development. For example, complete enumeration of the

decision spaces for the scenario and resource scheduling subproblems generates approximately $3.47x10^{37}$ and $2.95x10^{6}$ scheduling outcomes, respectively.

Another exact method, dynamic programming (DP), seemed particularly well suited to modeling the dynamic aspects of the scheduling subproblems. For example, it accommodates the sequential nature of the decision processes discussed previously: the flexible interactive training (FIT) method and the fixed and sequential training (FAST) method. Dynamic programming makes it possible to obtain partial resource scheduling solutions for any period of the scheduling horizon and for any allowable state condition by appealing to Bellman's *Principle of Optimality* (see Bertsekas, 1987). However, enumeration of the resource scheduling problem state space is (again) a problem. The state of system $I(t)$, the number of idle RED SAF in each period *t*, evolves according to Equation (8). $I(t)$ may take on 528 RED SAF values in each period of the scheduling horizon.[3] The varying duration of training scenarios (see Table 1) creates a situation where not all the information needed to make a scheduling decision in the current period can be summarized in the state variable $I(t)$. This can only be overcome by augmenting the state of the current period $I(t)$ with time-lagged information from the six previous periods (see Bertsekas, 1987). From this we estimate that the augmented state space for the reformulated problem contains approximately $1.15x10^{19}$ possible values making DP impractical for solving the problem.

[3]Based on the number of RED SAF required for conducting five simultaneous scenarios. These include combinations of three platoons for platoon-within-company context or three companies for a company-within-company context, and two additional platoon-level scenarios.

4. EXPERT SYSTEM DEVELOPMENT

Discussions of scheduling issues with CCTT and TREDS experts revealed important observations about scheduling CCTT training. First, the size of the scheduling problem's decision and state spaces prohibits timely generation of acceptable CCTT training schedules using manual methods. Second, the presence of human factors in tailoring the selection of training scenarios to the needs of the military unit and problem size made the use of exact solution methods impractical. Third, CCTT experts felt suboptimal solutions to the CCTT problem obtained by heuristic methods would be "good enough" for scheduling CCTT training. Finally, it was very important to make the scheduling system as user-friendly as possible by automating (computerizing) as many scheduling tasks as possible. Given these issues and concerns, a hybrid expert system approach was chosen for scheduling CCTT training (see Ignizio, 1991). The expert system would enable the system user to apply human factors in selecting scenarios during system consultation. Furthermore, heuristic scheduling programs could be automated within the hybrid expert system to generate acceptable training scenario and resource schedules in a timely manner. Development of the hybrid expert system prototype followed four sequential, overlapping phases. These were knowledge acquisition, knowledge representation, heuristic scheduling program development, and expert system implementation.

In the knowledge acquisition phase, scheduling knowledge was identified and acquired for the expert system. Production rules (IF-THEN statements) were used to represent expert system knowledge. Heuristic program development involved flowcharting the scheduling logic for scenarios and training resources. In the final phase,

rules for selecting and scheduling scenarios were then implemented using *Visual Basic®* that is compatible with TREDS development in *Microsoft® Windows®*. The resource scheduling heuristic program was also implemented in *Visual Basic®*, although this may also be programmed on other domains such as spreadsheets like *Microsoft® EXCEL®*. Figure 3 illustrates the architecture and the major components of the hybrid expert system prototype.

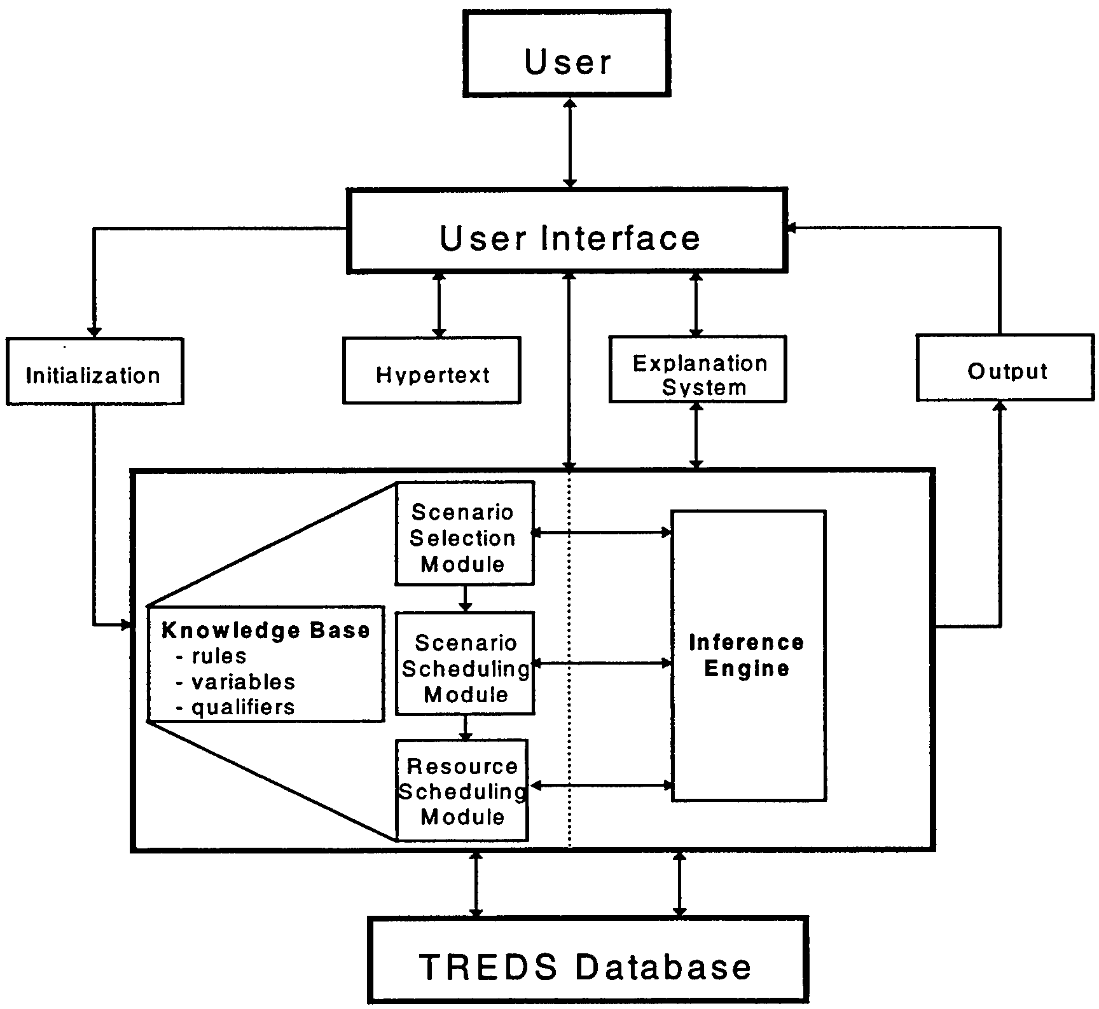

Figure 3. Expert system architecture.

4.1 PRELIMINARY RESULTS

The first CCTT site is not expected to be operational for at least a year or two from now. Therefore, real-world data for evaluating the quality of the expert system prototype schedules will not be unavailable for some time to come. Furthermore, the functions, capabilities, and performance of the expert system will certainly evolve as new CCTT scheduling requirements become known. Therefore, at this time, expert system prototype tests have been limited to verifying and validating the functionality of the prototype. Beyond this, preliminary system tests have been performed on the time it takes to generate a training scheduling using the system prototype and measuring the quality of the schedules generated based on a training system utilization performance measure.

According to SIMNET managers, training units typically spend from one to several days planning and scheduling SIMNET training. A set of ten realistic, feasible test cases was generated by the authors for evaluating the performance of the hybrid expert system prototype. The test cases involved scheduling training for a battalion, and combinations of lower echelon units, during a 1-day planning horizon. The training scenario events for each test case are given below in Table 6. The test cases involve scheduling scenarios consisting of three types of training events: movement to contact (MTC), attack (ATK), and defend (DEF), for three echelons of training units: battalion (BN), company (CO), and platoon (PLT). The type and number of training events, and the echelon of the training units scheduled throughout the training day are given in Column 1 of Table 6 for each test case.

Table 6. Scheduling results for a 1-day planning horizon.

Test Case Scenarios	Training Duration (hrs)	Total Scheduling Time (min)	Resource Utilization Value (Eqn 15)	Resource Utilization as a % of Utopian
1. 36 PLT events: 12 MTC, 12 ATK, 12 DEF	16	1:20	11.2	70%
2. 4 PLT events: 4 ATK 3 CO events: 2 MTC, 1 ATK 2 BN events: 1 MTC, 1 ATK2	13	1:31	11.57	89%
3. 12 PLT events: 4 MTC, 4 ATK, 4 DEF 12 CO events: 4 MTC, 4 ATK, 4 DEF	15	2:12	13.65	91%
4. 20 PLT events: 6 MTC, 7 ATK, 7 DEF 6 CO events: 2 MTC, 2 ATK, 2 DEF	13	2:08	10.53	81%
5. 9 PLT events: DEF 4 CO events: DEF 1 BN event: DEF	13	1:13	11.83	91%
6. 12 PLT events: MTC 4 CO events: ATK 1 BN event: DEF	15	1:44	12.90	86%
7. 24 PLT events: 8 MTC, 8 ATK, 8 DEF 4 CO events: 1 MTC, 1 ATK, 2 DEF	15	1:35	10.95	73%
8. 18 PLT events: 6 MTC, 6 ATK, 6 DEF 4 CO events: 2 MTC, 3 ATK, 3 DEF	15	1:24	11.25	75%
9. 7 PLT events: 2 MTC, 2 ATK, 3 DEF 5 CO events: 2 MTC, 2 ATK, 1 DEF 1 BN event: ATK	13	1:16	11.57	89%
10. 11 PLT events: 3 MTC, 3 ATK, 4 DEF 7 CO events: 2 MTC, 2 ATK, 3 DEF 1 BN event: MTC	12	1:23	11.40	95%

The scheduling results in Table 6 were obtained by the second author, highly skilled at using the scheduling system prototype. The scheduling system was evaluated

based on two performance measures. One was *total scheduling time*, in minutes, to generate the training scenario and resource schedules for each test case. Time measurement started when the system user began a scheduling session with the expert system, and stopped when the scheduling program displayed the feasible scenario and RED SAF schedule on the computer screen. The second performance measure, system resource utilization, is the ratio of RED SAF scheduled in each period to the number available for scheduling, summed over T periods of the planning horizon. Equation 15, Appendix A, gives the mathematical representation of the performance measure. The *utopian value* for the resource utilization measure in any period equals one. Therefore, the utopian value for resource utilization for the planning horizon is T. Column 3 of Table 6 gives resource utilization value for each test case. Column 4 expresses the resource utilization value as a percentage of the utopian value.

As shown in Table 6, the expert system prototype consistently generated CCTT training schedules in less than two minutes for each of test case considered. The average scheduling time for the scenarios was approximately 1:40 minutes with an average resource utilization value measured at approximately 84% of optimal for the 1-day planning horizon scheduling problem. This is a significant improvement over manual scheduling methods used previously by training units using the Fort Knox SIMNET training site. Furthermore, it is expected that an improved, fully developed expert system will further reduce the time required for scheduling training units for CCTT training and will also likely improve system utilization as well.

4.2 KNOWLEDGE ACQUISITION

An important step in expert system development was to identify the rules for selecting and scheduling training scenarios, and scheduling training resources. Since none of the CCTT sites are currently operational, it was not possible to acquire scheduling knowledge by direct observation of a real-world system or through interviews with CCTT experts. For prototype development, expert system knowledge came from the following sources: CCTT system documentation (US Army STRICOM, 1991, & US Naval Training Systems Center, 1992), and interviews of CCTT and TREDS developers at STRICOM and SIMNET experts at Fort Knox. The study team also made several trips to the SIMNET facility at Fort Knox to make first-hand observations of SIMNET training and scheduling procedures. During SIMNET site visits, the authors interviewed army personnel at battalion and below undergoing SIMNET training. From these interviews the study team identified training unit scheduling information, obtained ideas for representing scheduling knowledge, and noted human factors important to software system development.

Figure 4 depicts a high-level view of the CCTT scheduling problem decomposed into the 3 subproblems discussed in Section 2. In Step 1 the system user provides information about the training unit for scheduling. In Step 2 the user consults with the expert system to select training scenarios. In Steps 3 and 4 the heuristics schedule scenarios and training resources, respectively.

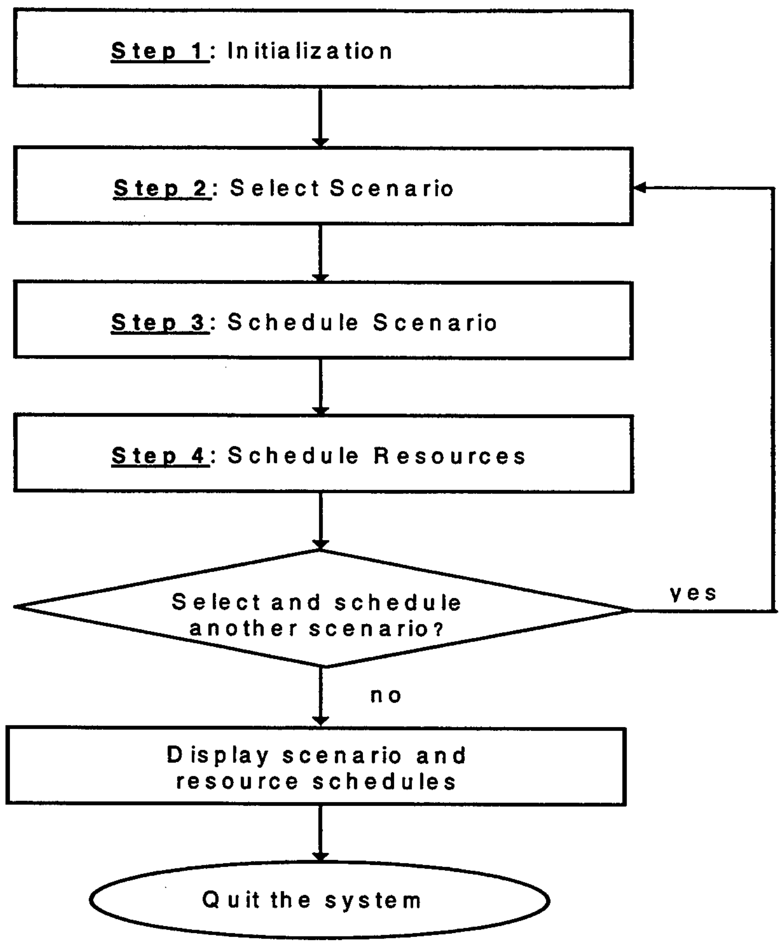

Figure 4. CCTT scheduling flowchart.

4.3 KNOWLEDGE REPRESENTATION

A number of methods may be used to represent knowledge within an expert system. These include production rules, semantic networks, frames, or object-attribute-value (OAV) triplets (see Ignizio, 1991, or Harmon, et.al., 1988). Of these, production rules seemed to be the most natural way to represent CCTT knowledge. It also provided flexibility in tailoring the expert system to the sequential nature of the scheduling decision process shown in Figure 4. Currently, the expert system prototype uses approximately 60 rules to select training scenarios and schedule resources. Figure 5

shows a partial decision tree framework for using information elicited from a system user during training scenario selection.

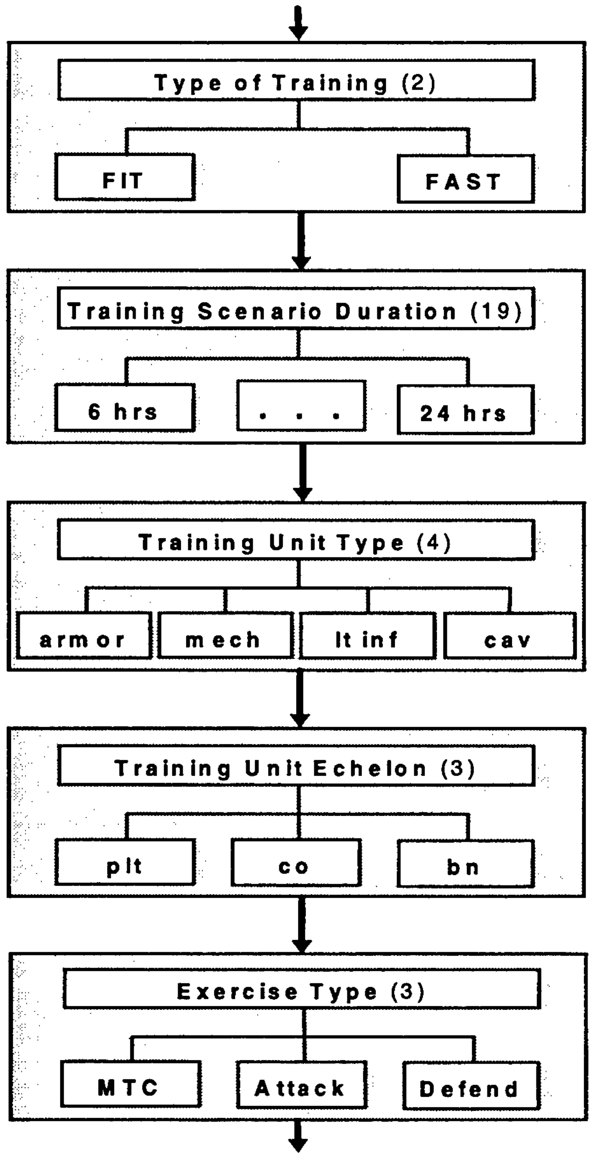

Figure 5. Partial decision tree for scenario selection.

The system user interacts with the expert system through a user interface module. The expert system knowledge base contains rules, variables, qualifiers and facts for conducting a scheduling session with the expert system, including the heuristics for scheduling scenarios and training resources. Facts used by the expert system during a session come from 3 sources. These are the TREDS database, the training unit (elicited

from the system user during consultation), and the heuristic programs during training scenario and resource scheduling. The inference engine processes rules and knowledge for generating a training schedule. Examples of user interface screens for scheduling with the expert system prototype are shown in Appendix C which takes the reader through an illustrative scheduling session.

4.4 HEURISTIC SCHEDULING PROGRAM

4.4.1 Scheduling Training Scenarios.

The heuristic program for scheduling training scenarios assumes that all training scenarios to be scheduled during the training day have been selected in advance by the system user. The heuristic program follows a forward recursion that starts in the first hour of the training day and schedules scenarios hour-by-hour to the end of the planning horizon. Within a period, lower echelon units are scheduled ahead of higher units. Within an echelon, scenarios for untrained units are scheduled ahead of partially trained units that, in turn, are scheduled before scenarios for fully trained units. An exchange technique (see Bertsekas, 1987) is used to schedule scenarios following the rules described above. The "best" suboptimal sequence of scheduling decisions $\{s_j^*(1),\ldots,s_j^*(t),\ldots,s_j^*(T-1)\}$ is the one that "maximizes" the value of the scenario schedule Φ_{π_S} according to equation (11) (see Section 4.3 for details). Figure 9 shows the logical flow of the heuristic rules for scheduling training scenarios.

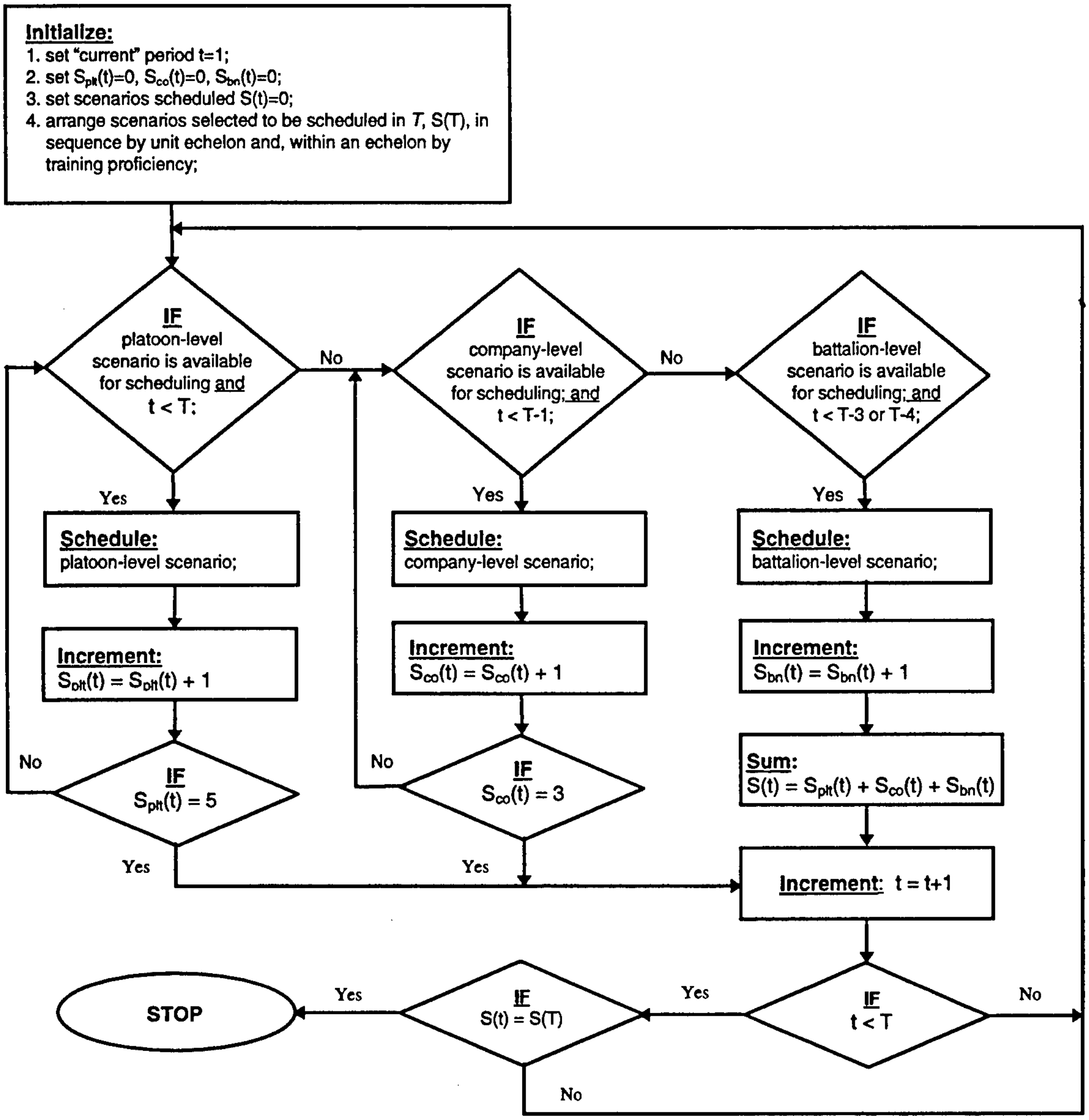

Figure 6. Heuristic rule processing for scheduling training scenarios.

This strategy generally leads to training scenarios of shorter duration for lower-echelon units being scheduled ahead of longer ones' for higher-echelon units. Similarly, scenarios requiring fewer RED SAF, such as those for untrained units, are generally scheduled before scenarios for trained units requiring higher numbers of RED SAF. In most cases, these rules result in lighter demand for RED SAF early in the horizon. In turn, this makes more RED SAF available later in the planning horizon for scheduling

higher-echelon, fully trained units. As a result, RED SAF shortfalls generally occur later in the scheduling horizon, if at all. This strategy contributes to the efficiency of the resource scheduling heuristics (see below) by requiring fewer iterations of the heuristic scheduling program to obtain an acceptable resource schedule, if one exists. However, it is noted that this result may be offset by scheduling more lower-echelon, untrained units that require fewer RED SAF early in the planning horizon.

4.4.2 Scheduling Training Resources.

The expert system estimates RED SAF strength $r_j(t)$ for each hour of the scheduling horizon based on the training scenario schedule and from information provided by the system user (see 2.4). The "best" suboptimal RED SAF decision sequence $\{r_j^*(1),...,r_j^*(t),...,r_j^*(T-1)\}$ from the heuristics is the one that maximizes training "quality" measured by the number of RED SAF scheduled for each training scenario. An alternate objective used here for achieving the same result used is to minimize the number of idle RED SAF $I(t)$ in each period according to equation (8) and Φ_{π_R} of (12). The sum of all RED SAF upper bound values $\bar{r}_j(t)$ of Table 3 over each hour *t* of the scheduling horizon establishes a *utopian* upper bound value for measuring the "quality" of competing feasible training schedules. The RED SAF scheduling heuristic is applied in 2 phases. These are referred to as the Phase 1 Heuristic (P1H) and Phase 2 Heuristic (P2H).

4.4.3 Phase 1 Heuristic (P1H).

The Phase 1 Heuristic determines whether a feasible resource schedule exists for the current training scenario schedule (see equation (6)). First, RED SAF strengths for all scenarios currently scheduled throughout the training day are temporarily fixed at the lower bound values $\underline{r}_j$. Next, the number of idle RED SAF $I(t)$ is computed for each of T hours. If $I(t) < 0$ in any hour t, then the current scenario schedule is *infeasible*. Accordingly, the current schedule must be modified by either substituting or deleting a scenario from the current hour t where $I(t) < 0$, or from an earlier period, to make more RED SAF available so that $I(t) \geq 0 \ \forall \ t$.

If a feasible resource schedule exists, where $I(t) \geq 0 \ \forall \ t \in T$, then the Phase 1 Heuristic continues according to the steps outlined below in Figure 10. At this point, RED SAF strengths for each scenario are reinitialized at the upper bound values. This may result in an initial, possibly infeasible RED SAF schedule for the current training scenario schedule where $I(t) < 0$ for some t. Next, the Phase 1 Heuristic makes a single <u>forward</u> pass through the planning horizon to schedule RED SAF in each period of the planning horizon. The RED SAF scheduling policy of Phase 1 is the sequence of training scenario and RED SAF strength decisions for each period t is specified by

$$\pi^1 = \begin{Bmatrix} s_j^1(1),\dots,\ s_j^1(t),\dots,\ s_j^1(T-1); \\ r_j^1(1),\dots,\ r_j^1(2),\dots,\ r_j^1(T-1) \end{Bmatrix}. \tag{14}$$

Superscript 1 denotes Phase 1.

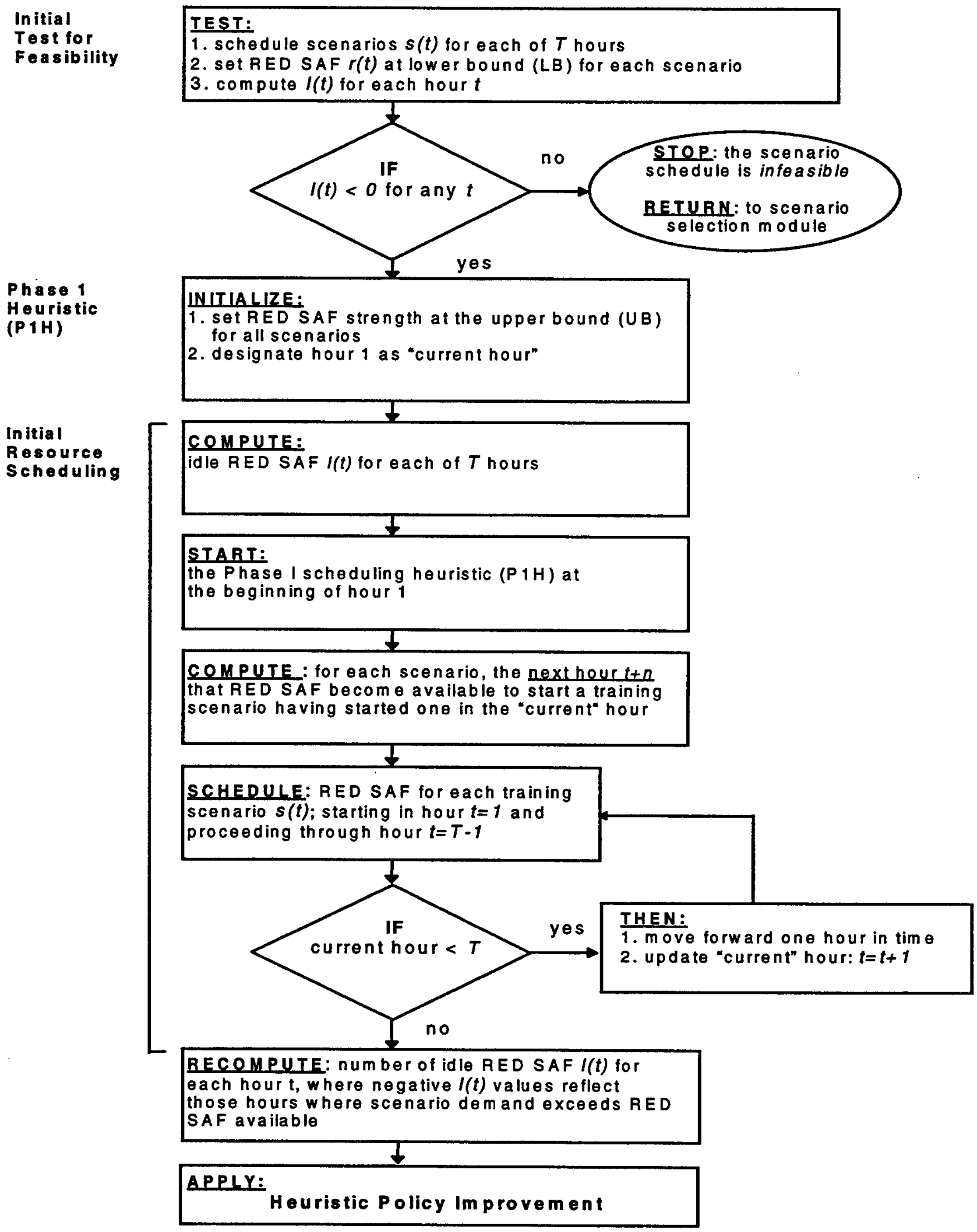

Figure 7. Phase 1 Heuristic for scheduling RED SAF.

4.4.4 Phase 2 Heuristic (P2H).

In any hour t, it is possible for previous scheduling decisions to create a RED SAF shortfall so that $I(t) < 0$. When this occurs, the training resource scheduling decisions

from previous periods must be sequentially revised, if possible, to make more RED SAF available to meet demand.

In Phase 2, a forward recursion sequentially revises the RED SAF strengths from Phase 1 hour-by-hour by making successive forward passes through the planning horizon to make $I(t) \geq 0$ in each period and then stops. These revisions are accomplished with an automated policy improvement step motivated by dynamic programming's policy improvement step (see Bertsekas, 1987) but modified to fit the unique structure of the CCTT resource scheduling problem. Automated policy improvement eliminates the need for an analyst to interactively schedule resources; a noteworthy improvement over existing manual, trial-and-error scheduling methods.

The Phase 2 Heuristic begins in the first period of the scheduling horizon and moves forward to each hour t where $I(t) < 0$. The heuristic then works sequentially backward through the planning horizon hour-by-hour to correct the RED SAF shortfall. Within a period starting in period t, RED SAF strength $r_j(t)$ is decreased by one step of size n at a time to the RED SAF lower bound $\underline{r}_j$, if necessary, until $I(t) \geq 0$. The changes to RED SAF are applied scenario-by-scenario beginning with scenarios for highly trained units and ending with scenarios for less well-trained units. The resource scheduling policy obtained at the completion of Phase 2 is given by

$$\pi^* = \left\{ \begin{matrix} s_j^*(1), \ldots, s_j^*(t), \ldots, s_j^*(T-1); \\ r_j^*(1), \ldots, r_j^*(2), \ldots, r_j^*(T-1) \end{matrix} \right\}. \tag{15}$$

$r_j^*(t)$ denotes the RED SAF strength decision for period t *given all previous decisions.*

π^* represents the "best" suboptimal sequence of training scenario and resource scheduling decisions obtainable using the heuristic methods described above. Figure 11 flowcharts policy improvement of the Phase 2 Heuristic.

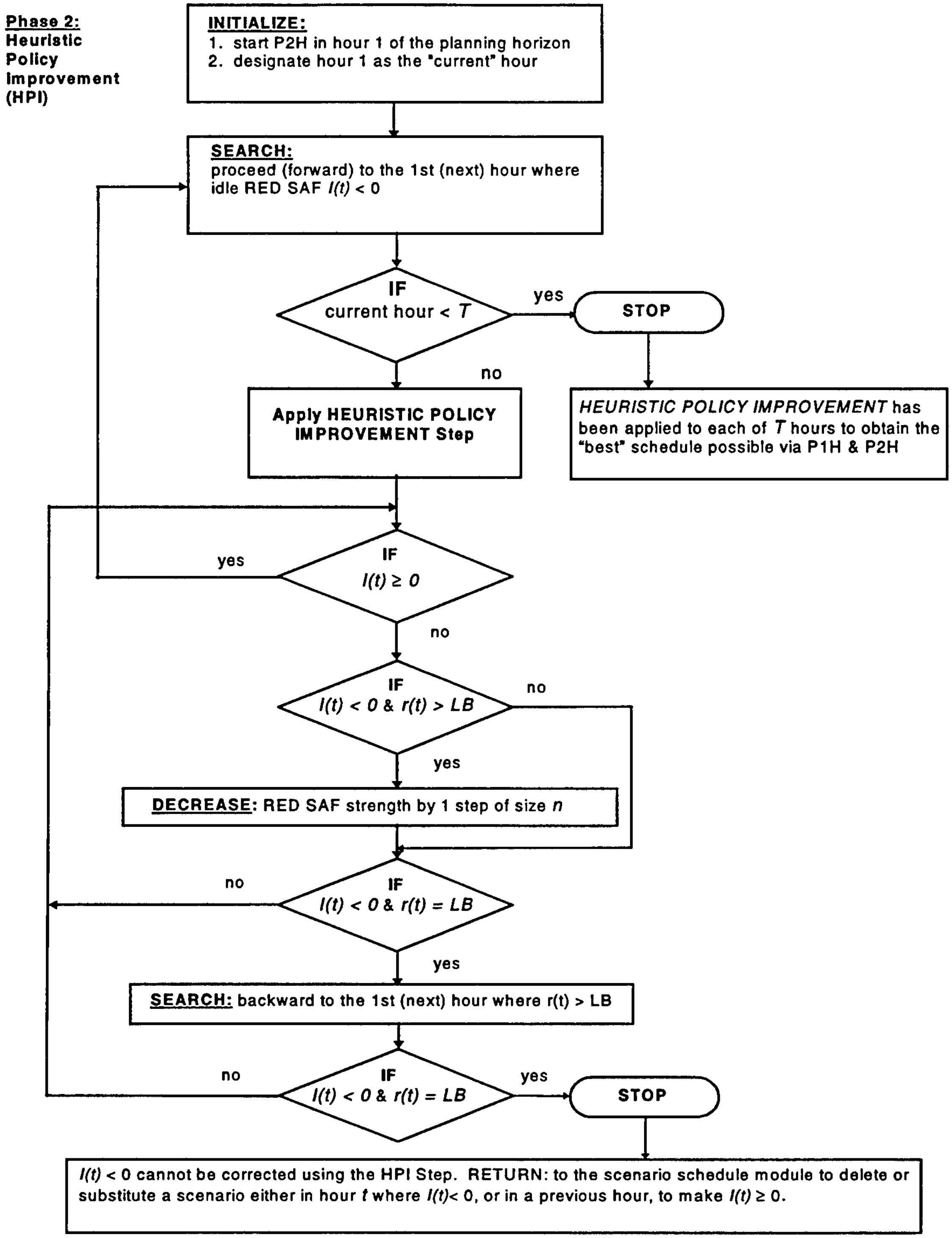

Figure 8. Phase 2 Heuristic Policy Improvement.

5. CONCLUSIONS

This technical report presents a hybrid expert system prototype for solving a complex scheduling problem of practical interest to the US Army for scheduling future training and training resources in the Close Combat Tactical Trainer environment. Specifically, the problem is to schedule training scenarios and training resources over a finite planning horizon of T stages.

5.1 MAJOR CONTRIBUTIONS

One contribution of this work is the mathematical formulation of the CCTT scheduling problem that accounts for important dynamics of the training process. Dynamic characteristics of the Close Combat Tactical Trainer system model include:

- varying training scenario duration;
- varying demand for training resources;
- varying levels of training resources per training scenario; and
- varying number of simultaneous training scenarios in any period.

In the real-world CCTT training problem, demand for training resources may be stochastic. Here however, training resource demand is known before making resource scheduling decisions for each of T time periods so that our problem is deterministic.

The scheduling objectives presented are consistent with measures used in practice for evaluating training. The scenario scheduling objective maximizes training "opportunity" as measured by the value of training scenarios scheduled during the training day to the training unit. The resource scheduling problem objective maximizes

"training quality," or alternatively, system utilization, as measured by the number of RED SAF assigned to each training scenario. An objective function for minimizing training program costs is also formulated.

The decision elements of the two scheduling subproblems reflect practical, real-world training decisions. For example, the schedule policy specifies the number and type of training scenarios to start in each hour of the training day and the number of RED SAF scheduled for each scenario.

A second major contribution is the development of a hybrid expert system prototype that implements the mathematical model of the CCTT problem given in Section 3. The expert system incorporates the scheduling and decision elements discussed previously featuring a two-phase heuristic program for scheduling scenarios and training resources.

5.2 POTENTIAL BENEFITS

Potential benefits of the expert system to the Army include the following.

- Improved unit readiness from effective training tailored to the needs of the unit.
- Improved resource utilization through better resource scheduling leading to tighter control of the CCTT training facility, system components, supplies, and personnel.
- Cost savings through improved training management.

5.3 SUGGESTIONS FOR FUTURE RESEARCH

Future research areas include the following.

- Incorporate other training resources into the mathematical model and the expert system.

- Extend the varying scheduling horizons for a single training day to multi-day planning horizons.

- Collect data for formulating a robust cost functional that reflect real-world costs.

- With the present version of TREDS, training scenarios are selected by the system user based on subjective judgment or personal reference. Research other measures for selecting scenarios. These include an index for comparing differences between the training event list (reflecting the unit's training needs) and training scenario events (reflecting training opportunity), scenario duration, or training resources.

- Investigate the effects on scheduling of stochastic demand for training resources, training delays due to system breakdowns and training unit difficulties, and recycling training units.

- Once CCTT systems are operational, incorporate real-world knowledge and rules into the expert system from first hand observations of CCTT operations.

- Improve the efficiency of the heuristic scheduling program for training scenarios and resources.

- Investigate the use of optimal solution methods such as dynamic programming for solving the CCTT scheduling problem.

- Extend the CCTT model and the expert system to other training facilities of the Combined Arms Tactical Trainer (CATT) family. It may be possible to apply the approach presented here perhaps to similar training programs of the Air Force, Navy, and Marines as well.

REFERENCES

Boldovici, J.A., & Bessemer, D.W. (1993). Training Research with SIMNET: Implications for Designing Research with Distributed Interactive Simulations. Alexandria, VA: US Army Research Institute.

Bertsekas, D. (1987). Dynamic Programming. Englewood Cliffs, NJ: Prentice-Hall.

Brown, D.E. & Scherer, W.T. (1995). Intelligent Scheduling Systems. Boston, MA: Kluwer Academic Publishers.

Clemen, R.T. (1990). Making Hard Decisions: An Introduction to Decision Analysis. Belmont, CA: Duxbury Press.

Crissey, M., Stone, G., Briggs, D., & Mollaghasemi, M. (1994). Training and Exercise Planning: Leveraging Technologies and Data. 16th Interservice/Industry Training Systems and Education Conference Proceedings, Orlando, FL, 6-12.

Harmon, P., Maus, R., & Morrissey, W. (1988). Expert Systems Tools and Applications. New York, NY: John Wiley & Sons, Inc.

Hollis, W., & Shoffner, W. (1992). Memorandum for Record appearing in the unpublished Users Conference on Distributed Interactive Simulations (DIS) Proceeding, Fort Monroe, VA.

Ignizio, J.P. (1991). Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems. New York, NY: McGraw-Hill, Inc.

Kandel, A. (1992). Fuzzy Expert Systems. Boca Raton, FL: CRC Press, Inc.

McGinnis, M.L., & Fernandez, E.G. (October 1994). Resource Scheduling for the United States Army's Basic Combat Training Program. 1994 IEEE International Conference on Systems, Man, and Cybernetics Proceedings. San Antonio, TX: IEEE Systems, Man, and Cybernetics Society, VOL. 1, pp.553-558.

McGinnis, M.L., & Fernandez, E.G. (December 1994). A Dynamic Programming Model for the Initial Entry Training Program of the United States Army. Proceedings of the 33rd IEEE Conference on Decision and Control. Lake Buena Vista, FL: IEEE Control Systems Society, VOL. 4, pp.3632-3633.

McGinnis, M.L., & Stone, G. (1995). Recent Army Initiatives to Develop Decision Support Systems (DSS) for Command and Control and Training Planning. 1995 International Conference on Industry, Engineering, and Management Systems (IEMS) Proceedings. Orlando, FL: University of Central Florida.

US Army Simulation, Training and Instrumentation Command (STRICOM). (1991). Statement of Work for the Close Combat Tactical Trainer (CCTT), Device Number 17-159. Orlando, FL: STRICOM.

US Army Training and Doctrine Command (TRADOC). (1994). TRADOC Pamphlet 525-5, 1994, Force XXI Operations: A Concept for the Evolution of Full-Dimensional Operations for the Strategic Army of the Early Twenty-First Century. Fort Monroe, VA: Headquarters TRADOC.

US Naval Training Systems Center. (1992). Systems Specifications for the Close Combat Tactical Trainer (CCTT), Device Number 17-159. Orlando, FL: US Naval Training Systems Center.

Yang, T. and Ignizio, J. (1987). Algorithm for the Scheduling of Army Battalion Exercises, Computers and Operations Research, VOL. 14, pp. 479-491.

APPENDIX A: SCHEDULING PROGRAM FOR THE CLOSE COMBAT TACTICAL TRAINER (CCTT).

This section discusses the scheduling program for the hybrid expert system. The program, written as pseudo-code, generates training schedules that are both *feasible* and *acceptable*. A feasible schedule is one that does not violate technological constraints of the CCTT system. An acceptable schedule is one that meets a scheduling performance measure identified in advance by the system user. Examples of acceptability criteria include system utilization and cost. For a discussion of CCTT performance measures, scheduling constraints, and related issues see Sections 2 and 3.

The scheduling program is based on the Fixed and Sequential Training (FAST) approach discussed in Section 2.2.3. However, it can be modified to work for the Fixed and Sequential Training (FIT) approach as well. Although discussed previously in Section 3, it is important to note here again that, in a mathematical sense, the program only generates "good" CCTT training schedules; not necessarily optimal ones. Therefore, the performance measures are also useful for comparing the quality of competing, suboptimal training schedules obtained from the scheduling program, or from some other source.

The scheduling program breaks down into three major, sequential tasks. The first task, Task 1, is initialization of the scheduling program. Next, in Task 2, training scenarios are dynamically scheduled, period by period, using a forward recursion. Following this, Task 3 generates a training resource schedule for the training scenario schedule of Task 2. CCTT scheduling policy π (see Section 3) is completely specified

by the training scenario and training resource scheduling decisions made in each period. The training scenario decision determines the type, number, and duration of training scenarios to start at the beginning of each period. The resource scheduling decision identifies the type and quantity of resources, RED SAF in this case, scheduled for each training scenario. We briefly discuss the three major tasks next. The significant steps within each task are also given.

A.1 PROGRAM INITIALIZATION

Program initialization involves (re)setting scheduling variables, parameters, and counters at initial values.

TASK 1. INITIALIZATION

STEP 1 Initialize: Set the "current" scheduling period $\hat{t}$ to $\hat{t} = 1$. This starts the scheduling program at the beginning of the planning horizon.

STEP 2 Initialize: Set variables for counting the number of training scenarios scheduled throughout the planning horizon to zero. Mathematically, $s_l(t) = 0 \;\; \forall \;\; t \in T$ and echelons $l \in L$. Here, L (currently) consists of the set $\{PLT, CO, BN\}$. As defined in Section 3, $s_l(t)$ denotes the number of training scenarios scheduled, by echelon l, during period t.

STEP 3 Initialize: Fix scheduling values for a battalion scenario (BN_VAL), company scenario (CO_VAL), and platoon scenario (PLT_VAL) in any period. Currently, the values for scheduling training scenarios by echelon l, also denoted l_VAL, are BN_VAL=32, CO_VAL=3, and PLT_VAL=2.

STEP 4 Initialize: Create a *mission essential task list* identifying key and essential training tasks for the military unit to accomplish during training.

A.2 SCHEDULING TRAINING SCENARIOS

The primary objective of scheduling training scenarios in Task 2 is to "maximize" the number scheduled in each of T planning periods. In turn, it is believed that this will generally lead to a highly utilized system. The training scenario scheduling program starts in the first period and works forward, one period at a time, until the end of the planning horizon. $T-1$ is the last period for making scheduling decisions.

Within each period, and throughout the planning horizon, the program schedules scenarios by unit echelon and unit training proficiency. In scheduling by unit echelon, we schedule platoon scenarios first, followed by company level scenarios, and then battalion scenarios. Within platoon and company echelons, we order (prioritize) scenarios based on the training proficiency of the platoons and companies undergoing training. In other words, scenarios for untrained units are scheduled first, followed by scenarios for partially trained units, with scenarios for trained units scheduled last. Ordering by training proficiency does not apply to battalion scenarios, however. The CCTT can only support one battalion level exercise at a time. Therefore, a battalion scenario excludes all others. This strategy for scheduling scenarios by echelon is consistent with the bottom-up approach to training generally followed by most Army units.

A scheduling equation, devised with the help of others (see Acknowledgments), is central to scheduling training scenarios. The scheduling equation developed is

$$V_l(\hat{t},\, \hat{t}-d_l(t)) \;=\; \sum_{l\in L} (l\_VAL)^{S_l(\hat{t})} + \sum_{d_l(t)\in D} \sum_{l\in L} (l\_VAL)^{S_l(\hat{t}-d_l(t))}\,. \qquad (14)$$

$l \in L$ denotes military unit echelons undergoing training and $d_l(t) \in D$ specifies the duration, in hours, of training scenarios (see Sections 2 and 3 for details). The scheduling values computed for each period, $V_l(\hat{t},\ \hat{t} - d_l(t))$, reflect the number and type of scenarios scheduled. Appendix B enumerates all combinations of training scenarios by echelon and the scheduling values $V_l(*)$ for each. The first term of Equation (14), $\sum_{l \in L} (l\_VAL)^{S_l(\hat{t})}$, accounts for scenarios scheduled in the current period $\hat{t}$. The second term, $\sum_{d_l(t) \in D} \sum_{l \in L} (l\_VAL)^{S_l(\hat{t} - d_l(t))}$, represents scenarios scheduled in previous periods $\hat{t} - d_l(t)$ that are still "busy" at the beginning of the current period.

As stated above in Task 1, the scheduling values for a single platoon (plt), company (co), and battalion (bn) training scenario in any period are PLT_VAL=2, CO_VAL=3, and BN_VAL=32, respectively. The scheduling equation, together with scheduling rules derived from CCTT technology constraints, are used to develop a scheduling policy for the day's training. Next, we give important rules used in Task 2 for scheduling training scenarios.

RULE 1 Scheduling a battalion scenario excludes all other scenarios.

RULE 2. In any period t, up to five training scenarios may be conducted simultaneously. Mathematically, this is denoted by

$$\sum_{l \in L} s_l(\hat{t}) + \sum_{d_l(t) \in D} \sum_{l \in L} s_l(\hat{t} - d_l(t)) \leq 5,$$

where $l \in L$ and $L = \{PLT, CO, BN\}$.

RULE 3. Up to three company scenarios may be scheduled, or ongoing, simultaneously during the same period. This is denoted by

$$s_{CO}(\hat{t}) + \sum_{d_l(t)\in D} s_{CO}(\hat{t} - d_l(t)) \leq 3$$

RULE 4. When scheduling five simultaneous scenarios in a period, if three are company level, then the remaining two must be platoon level scenarios.

RULE 5. When calculating $V_l(*)$, multiply training scenario scheduling values for training scenarios of the same type scheduled (busy) in the same period.[4]

RULE 6. When calculating $V_l(*)$, add training scenario scheduling values for training scenarios of different types scheduled (busy) in the same period.[5]

RULE 7. The maximum (upper bound), allowable scheduling value attainable by $V_l(\hat{t},\, \hat{t} - d_l(t))$ in any period of the planning horizon is 32. Mathematically, we require that

$$V_l(\hat{t},\, \hat{t} - d_l(t)) \leq 32.$$

TASK 2. TRAINING SCENARIO SCHEDULING PROGRAM

STEP 1 Select a set of training scenario candidates, S_J, from TREDS' training scenario data base that incorporates the training tasks from STEP 2 (see above). As explained in Section 2.2.4, J denotes training scenario categories enumerated from each combination of $\{k,l,m,n,o,p\}$.

STEP 2 Sort the set of training scenario candidates S_J by training unit echelon l. Training unit echelons l are platoon (PLT), company (CO), and battalion (BN) (see Table 1).

STEP 3 For each training scenario candidate $\tilde{s}_j$ to be scheduled in T, identify the training proficiency of the military unit(s) to conduct the training as either untrained (U), partially trained (P), or trained (T).

STEP 4 Order the training scenario candidates as follows: scenarios for untrained units ahead of scenarios for partially trained units. In turn, scenarios for untrained and partially trained units are ordered ahead of scenarios for trained units.

---

[4] For example, if scheduling three platoon scenarios in period t, then the total scheduling value for that period is PLT_VAL(t) = 2x2x2 = 8.

[5] For example, if scheduling two platoon and three company scenarios in period t, then the total scheduling value for that period is $2^2 + 3^3 = 31$.

STEP 5 Specify the *training unit context* for conducting each training scenario: platoon within platoon, platoon within company, platoon within battalion, company within company, company within battalion, and battalion within battalion (see Table 2).

STEP 6. TRAINING SCENARIO SCHEDULING PROGRAM.

A. Start the TRAINING SCENARIO SCHEDULING PROGRAM in period $\hat{t} = 1$ of the scheduling horizon.

B. **STOPPING RULES**

IF $\hat{t} = T$ **AND** platoon scenarios $s_{PLT}(t)$ are available,
THEN STOP. The end of the planning horizon has been reached and no more platoon scenarios may be scheduled. Otherwise,
CONTINUE.

IF $\hat{t} = T - 1$ **AND** company scenarios $s_{CO}(t)$ are available,
THEN STOP. The end of the planning horizon has been reached and no more company scenarios may be scheduled. Otherwise,
CONTINUE.

IF $\hat{t} = T - 1$ **AND** a 3 hour battalion scenario $s_{BN}(t)$ is available,
THEN STOP. The end of the planning horizon has been reached and a 3 hour battalion scenario may not be scheduled. Otherwise,
CONTINUE.

IF $\hat{t} = T - 3$ **AND** a 5 hour battalion scenario $s_{BN}(t)$ is available,
THEN STOP. The end of the planning horizon has been reached and a 5 hour battalion scenario may not be scheduled. Otherwise,
CONTINUE.

IF $\hat{t} = T - 4$ **AND** a 6 hour battalion scenario $s_{BN}(t)$ is available,
THEN STOP. The end of the planning horizon has been reached and a 6 hour battalion scenario may not be scheduled. Otherwise,
CONTINUE.

C. Compute the number of training scenarios, by echelon l, "busy" at the beginning of $\hat{t}$:

$$S_{PLT}(\hat{t},\, \hat{t}-d_{PLT}(t)) \;=\; s_{PLT}(\hat{t}) \;+\; \sum_{d_{PLT}(t)\in D} s_{PLT}(\hat{t}-d_{PLT}(t))$$

$$S_{CO}(\hat{t},\, \hat{t}-d_{CO}(t)) \;=\; s_{CO}(\hat{t}) \;+\; \sum_{d_{CO}(t)\in D} s_{CO}(\hat{t}-d_{CO}(t))$$

$$S_{BN}(\hat{t},\, \hat{t}-d_{BN}(t)) \;=\; s_{BN}(\hat{t}) \;+\; \sum_{d_{BN}(t)\in D} s_{BN}(\hat{t}-d_{BN}(t))$$

D. Compute the scheduling value for $\hat{t}$:

$$V_l(\hat{t},\, \hat{t}-d_l(t)) \;=\; \sum_{l\in L} (l\_VAL)^{S_l(\hat{t})} \;+\; \sum_{d_l(t)\in D}\; \sum_{l\in L} (l\_VAL)^{S_l(\hat{t}-d_l(t))}$$

E. **IF** $V_l(\hat{t},\, \hat{t}-d_l(t)) \;\geq\; 31$,
THEN no additional training scenarios may be scheduled in the current period $\hat{t}$. Let $\hat{t} = \hat{t}+1$, and
RETURN to STEP 6B. Otherwise,
CONTINUE.

F. **IF** $V_l(\hat{t},\, \hat{t}-d_l(t)) \;\leq\; 30$, **AND** $S_l(\hat{t}-d_l(t)) \;\leq\; 5$, **AND** a platoon training scenario $\tilde{s}_{PLT}(\hat{t})$ is available for scheduling in period $\hat{t}$ from the set of training scenarios S_J selected in STEP 1 of TASK 2,
THEN schedule $\tilde{s}_{PLT}(\hat{t})$ and
RETURN to STEP 6C. Otherwise
CONTINUE.

G. **IF** $V_l(\hat{t},\, \hat{t}-d_l(t)) \;\leq\; 29$, **AND** $S_{CO}(\hat{t}-d_l(t)) \;\leq\; 3$, **AND** a company scenario $\tilde{s}_{CO}(\hat{t})$ is available for scheduling in period $\hat{t}$ from the set of training scenarios S_J selected in STEP 1 of TASK 2,
THEN schedule $\tilde{s}_{CO}(\hat{t})$ and
RETURN to STEP 6C. Otherwise
CONTINUE.

H. **IF** $V_l(\hat{t},\, \hat{t}-d_l(t)) \;=\; 0$ **AND** a battalion training scenario $\tilde{s}_{BN}(\hat{t})$ is available for scheduling in period $\hat{t}$ from the set of training scenarios S_J selected in STEP 1 of TASK 2,
THEN schedule $\tilde{s}_{BN}(\hat{t})$ and
RETURN to STEP 6C.

A.3 SCHEDULING RED SAF TRAINING RESOURCES

In Task 3, the scheduling problem is to determine the type and quantity of resources to allocate against each scenario previously scheduled in Task 2. This is accomplished in a manner consistent with the original scheduling objective of maximizing system utilization via a resource scheduling objective that "minimizes" idle resources throughout the planning horizon.

A.3.1 SIMPLIFYING ASSUMPTIONS FOR THE SCHEDULING PROBLEM.

A simplified problem was chosen for implementation of the scheduling program during prototype development. First, we limited the problem to a single resource of importance to CCTT developers; namely, RED SAF. Second, we assumed RED SAF demand is known ahead of time, thereby making the RED SAF scheduling problem completely deterministic. Although this removes a very troublesome stochastic aspect of scheduling, determining the number of RED SAF to schedule in each period, or *RED SAF strength*, remains a formidable task. This is because RED SAF demand for each training scenario scheduled in Task 2 is not known exactly, but varies between upper and lower bounds (see Equation 1).

A.3.2 TRAINING RESOURCE DEMAND FOR RED SAF.

In any period t, RED SAF demand is measured by the number of RED SAF $r_j(t)$ required to execute the training scenarios $\tilde{s}_j(t)$ scheduled previously during Task 2. Demand varies by scenario type j, and within each type, as well. For prototype development, RED

SAF demand by scenario was elicited from training experts based on training unit echelon, training unit proficiency at mission essential tasks, and training unit contexts for conducting training. From this information, estimates of upper and lower bounds for RED SAF demand were established for each type of scenario. The demand values were then entered in easy-to-use (automated), computer lookup tables within the hybrid expert system prototype (see A.3.4 below).

A.3.3 TRAINING RESOURCE SCHEDULING DECISIONS FOR RED SAF.

Determining RED SAF strength for period *t* requires the following information: (1) The type and number of scenarios to start training in period *t*; and (2) the number of RED SAF available at the beginning of period *t* to start a training scenario. However, the number of RED SAF available to start a scenario in period *t* depends upon past RED SAF scheduling decisions. For example, if RED SAF scheduling decisions made prior to period *t* resulted in RED SAF strengths near the lower bound, then perhaps more scenarios might have been scheduled to start in those periods than would have started if RED SAF strengths had been nearer the upper bounds. This, in turn, leaves fewer RED SAF available to start scenarios scheduled in period *t*. In any period, it is possible for previous RED SAF strength decisions to cause a *RED SAF shortfall*. This occurs when the number of RED SAF is not sufficient to meet the minimal (lower bound) demand for RED SAF in *t* determined from the type and number of training scenarios scheduled in that period. In such cases, RED SAF strength decisions for previous periods must be sequentially revised (if possible) to correct the RED SAF shortfall.

A.3.4 SAF SCHEDULING GUIDELINES.

For prototype development, CCTT training experts established fixed BLUE SAF strengths for each training scenario category *j* based on doctrine and CCTT system technology constraints (see Table 3 in Section 2). For this reason, BLUE SAF strength is not a decision element of the resource scheduling problem. The main guidelines followed in scheduling RED SAF are highlighted below.

- RED SAF strength varies by training scenario category *j*, and between upper and lower bounds within a training scenario category, as well.
- For prototype development, CCTT training experts estimated "average" RED SAF strengths that a partially trained BLUE forces might expect to face in a real-world military operation. The "average" RED SAF values established baselines by training scenario type *j* for scheduling RED SAF. The RED SAF upper and lower bounds were then computed as percentages of the baseline values for fully trained and untrained BLUE forces, respectively.
- RED SAF upper bound $\bar{r}_j$ represents a full strength RED force that a fully trained blue armored or mechanized force might be expected to face in a force-on-force engagement during a real-world military operation.
- RED SAF lower bound $\underline{r}_j$ reflects a minimal strength RED force that an untrained blue force might be expected to face in a force-on-force engagement.

A.3.5 RED SAF SCHEDULING PERFORMANCE MEASURE.

As discussed above, the objective of resource scheduling for prototype development was to achieve a highly utilized CCTT training system. The **R**esource scheduling performance measure Φ_{π_R} formulated for this objective is the ratio of RED SAF scheduled in each

period to RED SAF available, summed over the entire planning horizon. Mathematically, this is given by

$$\Phi_{\pi_R} = \sum_{t=1}^{T} \left[\frac{\sum_{j \in J} r_j(t)}{I(t) + \sum_{j \in J} r_j(t)} \right]. \tag{15}$$

The *utopian value* for Φ_{π_R} in any period is one. Therefore, the utopian value for the entire planning horizon equals T, which is not attainable, in general.

A.3.6 TRAINING RESOURCE SCHEDULING PROGRAM.

Our program schedules RED SAF in three phases. First, Phase 1 determines whether or not a *feasible* training resource schedule exists for the training scenario schedule generated in Task 2, or from some other source. Assuming one exists, then Phase 2 uses a heuristic policy iteration procedure to generate a "good" initial RED SAF schedule for the training scenario schedule. In Phase 3, the scheduling program improves the scheduling policy from Phase 2 using a policy improvement procedure that makes multiple backward passes through the planning horizon until no further improvements to the RED SAF scheduling policy are possible.

Phase 1: Determining if a feasible RED SAF schedule exists.

In Phase 1, RED SAF demand is fixed at lower bounds $\underline{r}_j$ for each scenario $\tilde{s}_j(t)$ scheduled throughout the planning horizon. The resource scheduling program then makes a single, forward pass through the planning horizon scheduling idle RED SAF $I(t)$ to meet demand in each period.

As discussed previously, demand for idle RED SAF $I(t)$ is measured by the number of RED SAF needed to accomplish a training scenario. A balance equation computes idle RED SAF $I(t+1)$ at the beginning of each period (see Equation 8). Idle RED SAF are always carried forward from the end of the previous period. The balance equation also accounts for RED SAF that become idle having just completed training scenarios that started either two, three, five, or six hours earlier. Negative values of $I(t)$ represent backlogged demand; an extension to the problem that is not considered here.

If $I(t) \geq 0 \;\; \forall \;\; t \in T$, then a *feasible* resource schedule is guaranteed to exist for the training scenario schedule. However, from a training system viewpoint, scheduling RED SAF at lower bound values $\underline{r}_j$ throughout the planning horizon obviously leaves the system under utilized. Assuming a feasible RED SAF schedule exists, then the objective of Phase 2 is to generate a "good" initial scheduling policy for RED SAF (see below).

On the other hand, if $I(t) < 0$ for some $t \in T$, then a RED SAF shortfall exists that cannot be corrected by adjusting RED SAF strength since all values are fixed at $\underline{r}_j$. In this case, a feasible training resource schedule cannot exist, and the current training scenario schedule is said to be *infeasible*.

At this point, the system user either quits the system or returns to Task 1. Returning to Task 1, the system user either modifies the current training scenario schedule to correct training resource shortfalls in all periods where $I(t) < 0$, or generates a new training scenario schedule. Two options exist for modifying a scenario schedule so that $I(t) \geq 0 \;\; \forall \;\; t \in T$. One, replace one or more training scenarios scheduled to start in *t*, or in earlier periods, with scenarios that require fewer RED SAF. Two, delete one or more

training scenarios from the schedule to make sufficient RED SAF available, thereby covering the shortfall(s).

Phase 2: Obtaining an initial feasible RED SAF schedule.

First, RED SAF strength is (re)initialized for all $\tilde{s}_j(t),\ t \in T,$ at upper bound values $\bar{r}_j$. If $I(t) \geq 0 \ \forall \ t \in T$, then system utilization cannot be improved and the current Resource scheduling policy π_R is the best one achievable, denoted π_R^*. Generally, however, meeting RED SAF demand by setting strengths at upper bound values will cause a RED SAF shortfall in some period(s) so that $I(t) < 0$.

The scheduling program corrects shortfalls via a scheduling policy iteration procedure. The policy iteration procedure makes a single forward pass through the planning horizon starting in $t = 1$. It proceeds to the first hour $\hat{t}$ where $I(\hat{t}) < 0$. The policy iteration procedure then decreases RED SAF strength $r_j(t)$ for each training scenario scheduled to start in $\hat{t}$, in turn, by one step of size *n*, where $n \in \left\{1, \ldots, \bar{r}_j - \underline{r}_j + 1\right\}$. The size of *n*, an integer valued scheduling parameter set by the system user, is chosen based on tradeoffs between the need for a high quality scheduling policy and the time it takes to obtain one. For example, selecting $n = 1$ may lead to the highest quality scheduling policy attainable since it guarantees a "tight" problem feasibility constraint, $I(t) = 0$, in each period where possible. However, this improvement in quality comes at a cost in time to perform the additional iterations of the scheduling program.

The policy iteration step continues until either a decrease in RED SAF makes $I(\hat{t}) \geq 0$, or RED SAF strengths for each scenario $\tilde{s}_j(\hat{t})$ scheduled in that period reach their lower bound values. If $I(\hat{t}) < 0$ and $r_j(\hat{t}) = \underline{r}_j$ for all $\tilde{s}_j(\hat{t})$, then the scheduling program moves sequentially back in time, one period at a time, applying the policy iteration procedure at each stage until $I(\hat{t}) \geq 0$.

The instant the policy iteration procedure causes $I(\hat{t}) \geq 0$, then the shortfall has been corrected in $\hat{t}$, and the policy iteration procedure stops. The scheduling procedure then moves forward in time from $\hat{t}$ to the next hour where another $I(\hat{t}) < 0$. The policy iteration procedure is again applied to correct the RED SAF shortfall. This continues until the scheduling program reaches the end of the planning horizon.

Phase 3: Improving the initial feasible RED SAF schedule from Phase 2.

Phase 3 uses the initial feasible schedule from Phase 2 as its starting point. This RED SAF scheduling policy is revised, period by period, using a Heuristic Policy Improvement Procedure that works sequentially <u>backward</u> through the planning horizon until no further improvements to the objective function are possible. The final resource scheduling policy, obtained at the completion of Phase 3, is the one that "maximizes" the RED SAF scheduling objective Φ_{π_R}, and is given by

$$\pi_R^* = \left\{ r_j^*(1), \ldots, r_j^*(T) \right\}.$$

The heuristic policy improvement procedure starts in period T of the planning horizon and proceeds <u>backward</u> in time to each period $\hat{t}$ where $I(\hat{t}) > 0$ <u>and</u> $r_j(\hat{t}) < \overline{r}_j$.

RED SAF strength $r_j(\hat{t})$ is increased for each scenario, one scenario at a time and by one step of size n at a time, until $I_j(\hat{t}) < 0$. After each RED SAF increase, the scheduling program checks each period of the planning horizon within $\{\hat{t},...,T\}$ to ensure the last iteration of the Heuristic Policy Improvement Procedure did not cause the current policy to become infeasible, that is, $I(\tilde{t}) < 0$ for some $\tilde{t} \in \{\hat{t},...,T\}$. The first time that an increase of $r_j(\hat{t})$ causes $I(\tilde{t}) < 0$ in some period $\tilde{t} \in \{\hat{t},...,T\}$, then the last (most recent) iteration is undone by decreasing RED SAF strength in $\hat{t}$ by one step of size n to make $I(\tilde{t}) \geq 0$. This process is repeated, step by step and period by period, until no further improvements can be made to the RED SAF strength policy. This results in the "best" scheduling policy π_R^* obtainable via the heuristic procedure of Phase 3.

In summary, the heuristic procedures presented here are based on the mathematical formulation of the CCTT training problem given in Section 3. The procedures are designed to methodically correct RED SAF shortfalls throughout the planning horizon, and improve the RED SAF scheduling policy so that the system is highly utilized. The step by step procedure for implementing the RED SAF scheduling program is given next.

TASK 3. TRAINING RESOURCE SCHEDULING PROGRAM

PHASE 1: DETERMINING EXISTENCE OF A FEASIBLE RED SAF SCHEDULE.

1. Initialize: Set the "current" scheduling period $\hat{t}$ to $\hat{t}=1$, $I(0)=\overline{R}$, and $r_j(t)=\underline{r}_j\ \forall\ (j,t)$ so that RED SAF demand for each scenario is fixed at RED SAF lower bound values.

2. Get the duration $d_j(\hat{t})$ for each scenario $\tilde{s}_j(\hat{t})$ scheduled in $\hat{t}$.

3. Compute the <u>next</u> period where RED SAF that begin a training scenario in $\hat{t}$ will (again) be available (idle) to start another training scenario as $\hat{t}+d_j(\hat{t})$.

4. Get $\underline{r}_j(\hat{t})$ for each scenario $\tilde{s}_j(\hat{t})$ scheduled in $\hat{t}$.

5. Schedule $\underline{r}_j(\hat{t})$ for scenario $\tilde{s}_j(\hat{t})$ in $\hat{t}$, and schedule $\underline{r}_j(\hat{t})$ to be available (idle) for scheduling (again) in $\hat{t}+d_j(t)$.

   IF period $\hat{t}+d_j(t)\le T$;
   THEN let $\hat{t}=\hat{t}+1$, move forward one period in the planning horizon to $\hat{t}+1$, and return to STEP 2. Otherwise
   CONTINUE.

   IF period $\hat{t}+d_j(t)>T$;
   THEN the RED SAF SCHEDULING PROGRAM has completed scheduling for the entire planning horizon.
   CONTINUE.

6. Compute: $I(\hat{t}+1) = I(\hat{t}) + \sum_{j=1}^{J} \sum_{d_j(t)\in D} r_j(\hat{t}-d_j(t)) - \sum_{j=1}^{J} r_j(\hat{t})\quad \forall\ (j,t)$.

   IF $I(\hat{t}+1)\ge 0\ \forall\ t\in T$;
   THEN a *feasible* RED SAF schedule exists for the current training scenario schedule.
   GOTO Phase 2, otherwise
   CONTINUE.

IF $\exists\ \hat{t} \in \{1,...,T\} \ni I(\hat{t}+1) < 0$ AND $r_j(t) = \underline{r}_j \ \forall\ t \in T$;

THEN a feasible RED SAF schedule does not exist for the current training scenario schedule.

STOP.

PHASE 2: GENERATING AN INITIAL FEASIBLE RED SAF SCHEDULE VIA HEURISTIC POLICY ITERATION PROCEDURE

7. Initialize: Set the "current" scheduling period $\hat{t}$ to $\hat{t} = 1$ and $r_j(t) = \bar{r}_j \ \forall\ (j,t)$ so that RED SAF demand for each scenario is fixed at RED SAF upper bound values.

8. (Re)compute:

$$I(\hat{t}+1) = I(\hat{t}) + \sum_{j=1}^{J} \sum_{d_j(t)\in D} r_j(\hat{t} - d_j(t)) - \sum_{j=1}^{J} r_j(\hat{t}) \quad \forall\ (j,t).$$

IF there are no periods where $I(\hat{t}+1) < 0$ by stage $\hat{t}+1 = T$;

THEN STOP. The current feasible RED SAF scheduling policy π_R is the best one attainable, and is denoted π_R^*. Otherwise

CONTINUE.

9. Proceed sequentially <u>forward</u> through each period of the planning horizon to the first (or next) period $\hat{t}$ with a RED SAF shortfall, such that, $I(\hat{t}+1) < 0$.

10. Designate $\hat{t}$ as the <u>s</u>hort<u>f</u>all period $\hat{t}_{SF}$.

IF period $\hat{t}_{SF} + 1 > T$;

THEN STOP. The POLICY ITERATION PROCEDURE for Phase 2 has been applied to each period of the planning horizon resulting in an *initial feasible resource schedule* π_R^2, denoted by the superscripted 2. Proceed to Phase 3. Otherwise

CONTINUE.

11. RED SAF POLICY ITERATION PROCEDURE:

IF $I(\hat{t}_{SF}+1) = 0$;

THEN return to STEP 9. Otherwise

CONTINUE.

IF $I(\hat{t}_{SF}+1)<0$ **<u>and</u>** $r_j(\hat{t}_{SF})>\underline{r}_j$;

THEN iteratively decrease RED SAF strength by one step of size $\boldsymbol{n}$, for each scenario $\tilde{s}_j(\hat{t}_{SF})$ one scenario at a time. Replace $r_j(\hat{t}_{SF})$ by $r_j(\hat{t}_{SF})-n$, and return to STEP 8. Otherwise

CONTINUE.

IF $I(\hat{t}_{SF}+1)<0$ **<u>and</u>** $r_j(\hat{t}_{SF})=\underline{r}_j$ **<u>and</u>** $\hat{t}_{SF}-i\geq 1$, where $i\in\left\{1,\ldots,\hat{t}_{SF}-1\right\}$;

THEN proceed sequentially <u>backward</u> from $\hat{t}_{SF}+1$ to the first (or next) period $\hat{t}_{SF}-i$, where $i\in\left\{1,\ldots,\hat{t}_{SF}-1\right\}$, and where $r_j(\hat{t}_{SF})>\underline{r}_j$, such that i takes on values sequentially beginning with $i=1$. Return to STEP 11. Otherwise

CONTINUE.

IF $\hat{t}_{SFW}-i<1$ **<u>and</u>** $r_j(\hat{t})=\underline{r}_j \;\; \forall \;\; t\in\{1,\ldots,\hat{t}\}$;

THEN STOP. No further adjustments to RED SAF strengths are possible, and the RED SAF shortfall in period $\hat{t}_{SF}+1$ has been corrected. Return to STEP 9.

PHASE 3: HEURISTIC POLICY IMPROVEMENT PROCEDURE

12. Start the HEURISTIC POLICY IMPROVEMENT PROCEDURE in period T.

13. Proceed sequentially <u>backward</u> through the planning horizon from T to the first (or next) period $\hat{t}$, where $I(\hat{t})>0$ **<u>and</u>** $r_j(\hat{t})<\bar{r}_j$ for some scenario $\tilde{s}_j(\hat{t})$.

IF the current period $\hat{t}<1$;

THEN STOP. The HEURISTIC POLICY IMPROVEMENT PROCEDURE has been applied to the entire planning horizon. The scheduling policy for Phase 3, π_R^3, is the best one obtainable using the procedure, and denote the policy as π_R^*. Otherwise

CONTINUE.

14. RED SAF POLICY IMPROVEMENT STEP

A. Increase RED SAF strength for each scenario $\tilde{s}_j(\hat{t})$, in turn, by one step of size n in period $\hat{t}$ from $r_j(\hat{t})$ to $r_j(\hat{t})+n$.

B. Verify feasibility of the schedule. Check each period $\tilde{t} \in \{\hat{t},...,T\}$, sequentially, to ensure $I(\tilde{t}) \geq 0$.

IF $I(\tilde{t}) \geq 0 \quad \forall \; \tilde{t} \in \{\hat{t},...,T\}$;

THEN return to period $\hat{t}$ in the model, and return to STEP 13 of the scheduling program. Otherwise
CONTINUE.

IF $\exists \; \tilde{t} \in \{\hat{t},...,T\} \ni I(t) < 0$;

THEN return to period $\hat{t}$ in the model, undo the last increase of $r_j(\hat{t})$ by one step of size n to make $I(\tilde{t}) \geq 0$, and
RETURN to STEP 13.

APPENDIX B: POSSIBLE EXERCISE COMBINATIONS FOR A 12-HOUR TRAINING DAY.

Table 7 enumerates all of the possible combinations of training scenario events that can be scheduled during a 12 hour planning horizon. Table 7 provides training units with an automated guideline for CCTT training based on a reasonable determination of how many events may be scheduled during their time in the CCTT. Each row represents the total number of each type of exercise. The combinations account for the scheduling rules given in Appendix A.

Table 7. Training exercise combinations enumerated for a 12 hour training day.

Battalion				
Defend	Attack	Movement to Contact	Company	Platoon
2	0	0	0	0
1	1	0	0	0
0	2	0	0	0
1	0	2	0	0
0	1	2	0	0
0	0	4	0	0
1	0	0	6	6
1	0	0	5	7
1	0	0	4	9
1	0	0	3	10
1	0	0	2	12
1	0	0	1	13
1	0	0	0	15
1	0	1	3	2
1	0	1	2	3
1	0	1	1	4
1	0	1	0	5
0	1	0	6	6
0	1	0	5	7
0	1	0	4	9
0	1	0	3	10
0	1	0	2	12
0	1	0	1	13
0	1	0	0	15
0	1	1	3	4
0	1	1	2	6
0	1	1	1	8

Battalion				
Defend	Attack	Movement to Contact	Company	Platoon
0	1	1	0	10
0	0	3	1	4
0	0	3	0	5
0	0	2	6	6
0	0	2	5	7
0	0	2	4	9
0	0	2	3	10
0	0	2	2	12
0	0	2	1	13
0	0	2	0	15
0	0	1	9	8
0	0	1	8	9
0	0	1	7	11
0	0	1	6	12
0	0	1	5	13
0	0	1	4	15
0	0	1	3	16
0	0	1	2	17
0	0	1	1	19
0	0	1	0	20
0	0	0	12	12
0	0	0	11	13
0	0	0	10	15
0	0	0	9	16
0	0	0	8	18
0	0	0	7	19
0	0	0	6	21
0	0	0	5	22
0	0	0	4	24
0	0	0	3	25
0	0	0	2	27
0	0	0	1	28
0	0	0	0	30
0	2	0	0	5

APPENDIX C: ILLUSTRATIVE SESSION USING THE CCTT HYBRID EXPERT SCHEDULING SYSTEM.

The Close Combat Tactical Trainer Scheduling System has been designed from the maneuver battalion commander's viewpoint. The illustrated session that follows is based on a hypothetical Task Force, 3/4 Infantry. It is a mechanized balanced task force with 2 armored and 2 mechanized company teams. The unit's task organization is as follows:

<u>Task Force 3/4 Infantry</u>

<u>Team A 3/4 Infantry</u>	<u>Team B 2/3 Armor</u>
1/A 3/4 Infantry	2/A 3/4 Infantry
1/B 2/3 Armor	2/B 2/3 Armor
3/A 3/4 Infantry	3/B 2/3 Armor
<u>Team B 3/4 Infantry</u>	<u>C 2/3 Armor</u>
1/C 2/3 Armor	2/C 2/3 Armor
2/B 3/4 Infantry	3/C 2/3 Armor
3/B 3/4 Infantry	1/B 3/4 Infantry

In the past, when planning for training, unit leaders manually scheduled and sequenced training units and tasks. The CCTT Scheduling System automates the tasks for scheduling training in the CCTT facility. It is designed to be implemented within TREDS, however, it may also stand alone as a *Windows*®-based program.

The system user is welcomed to the CCTT scheduling system by the **Welcome Screen** whenever starting a consultation. The **Welcome Screen** includes the main menu selection bar for performing all scenario and training resource scheduling.

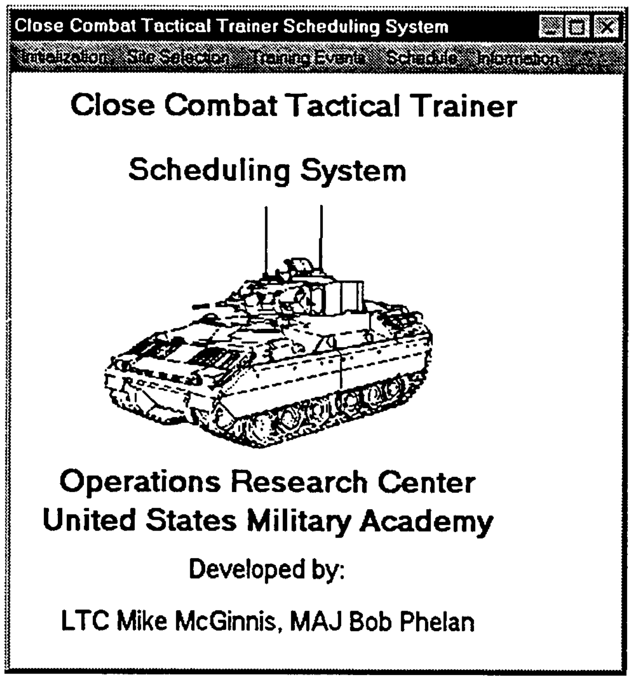

Figure 9. Welcome Screen.

The main menu options on the **Welcome Screen** are arranged according to the major scheduling steps, initialization through scheduling, from left to right. First, the program checks for the unit's task proficiency database. This database may be programmed in either *Access*® or *FoxPro*®. The database name and location are stored in the ***.ini*** file, and used by the program during the scheduling session. If the program can not find the database, it immediately informs the system user. Figures 10 and 11, below, depict the two-step process for updating the task proficiency database.

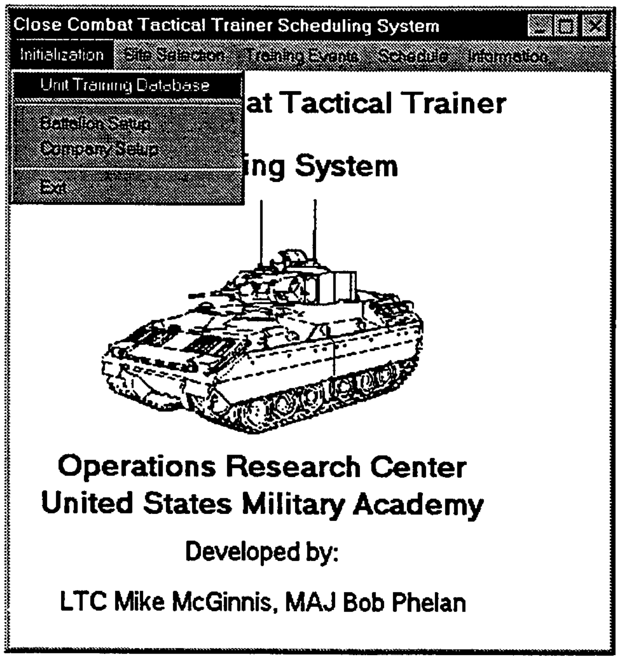

Figure 10. Step 1: Select database.

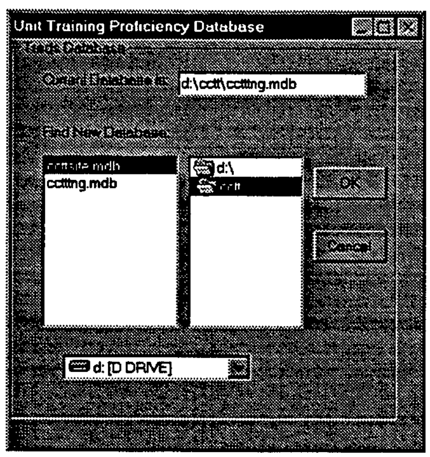

Figure 11. Step 2: Change database location.

Next, the system user initializes the Battalion/Task Force organization within the scheduling system. This is accomplished via a dropdown option selected from *Initialization* on the **Welcome Screen** and is illustrated below in Figure 12. When the *Battalion Setup* choice is entered, the user updates the Battalion/Task Force name,

availability, and the "active" sub-units. In the scheduler prototype, only three types of scenarios are available for scheduling: *Movement to Contact, Attack* and *Defend.* Eventually, TREDS may include other tasks besides these. Changes on this screen are automatically saved to the training proficiency database. If the *Entire Battalion will Train* item is marked, then all subordinate units will be considered available for training. Checking either *Battalion will train, not all subordinate units available*, or *Battalion will not train - only subordinate units* allows the system user to change availability of lower echelon units. Once completed, the user clicks *Finished.*

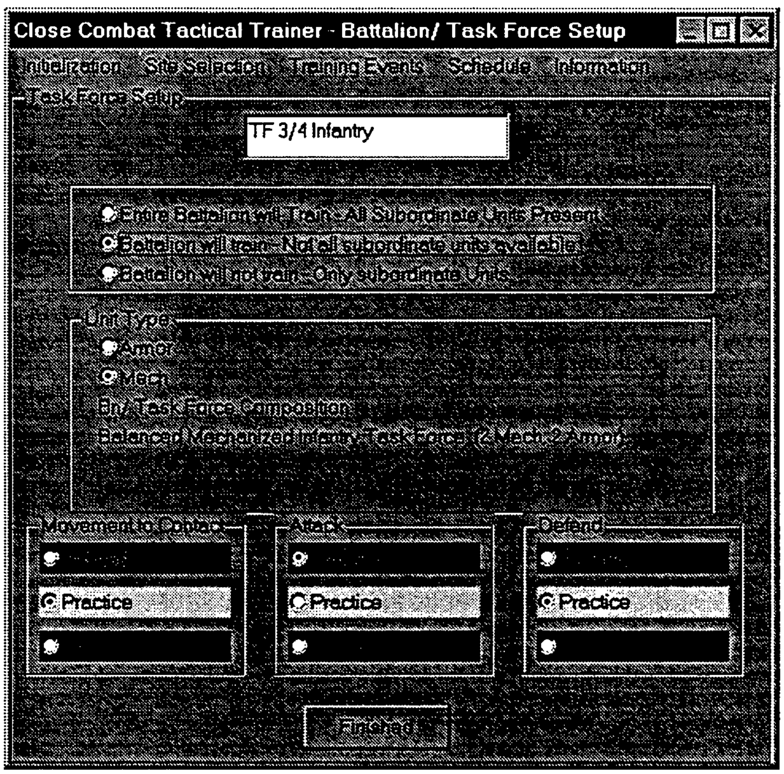

Figure 12. Battalion Setup.

The next step involves activating company teams and platoons within the task force for scheduling. When *Company Setup* is selected under *Initialization*, a card catalog-like

screen appears (see Figure 13). Company/Teams are listed along the left edge with subordinate platoons to the right of their respective companies. Clicking any of the unit tabs brings up information on that unit. The company screen gives the user an opportunity to change unit designation, availability, composition, and training proficiency by training task(s). This screen also provides the user with the opportunity to prioritize the units for scheduling using a scale from 1 to 5; 1 being the highest priority. This establishes a priority for scheduling units within the same echelon based on training proficiency. Subordinate unit availability is specified by the appropriate entry on the Company/Team data screen (see below).

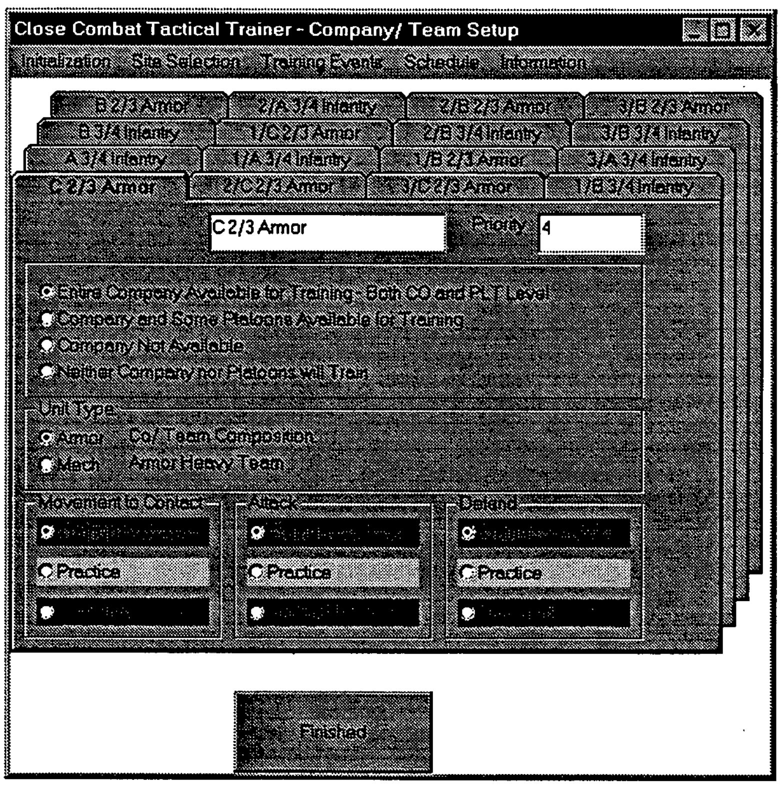

Figure 13. Company Setup.

In Figure 14, the *Platoon Leadership Available* option allows the platoon to be made "unavailable" for training. However, the platoon leader will be scheduled to participate in company and battalion exercises. This is similar to a real-world Tactical Exercise Without Troops (TEWT) and might be used, when necessary, to reduce the number of manned modules required for conducting company exercises.

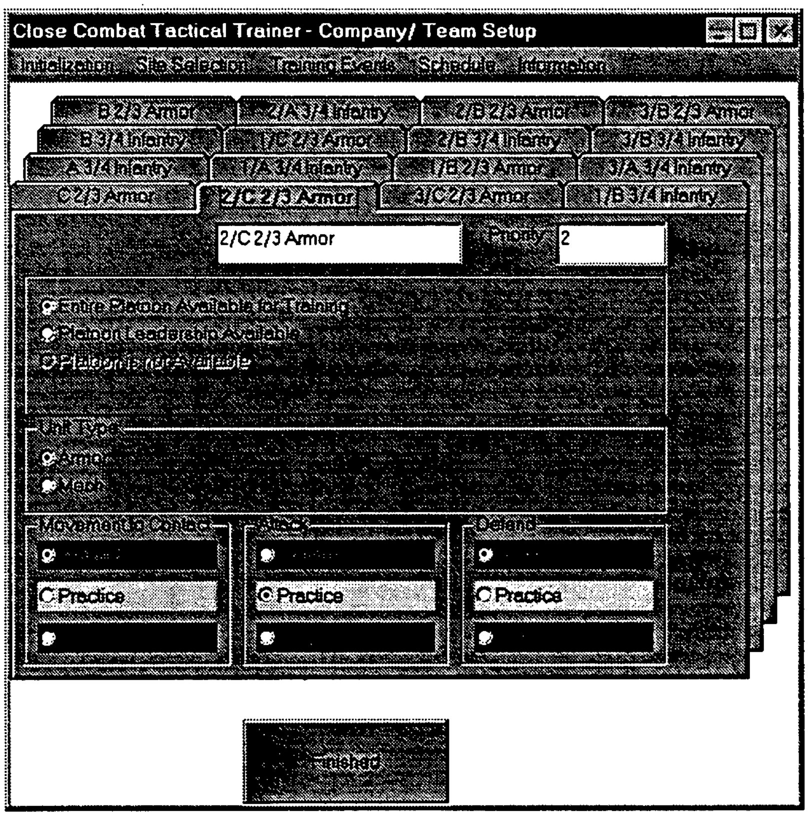

Figure 14. Platoon Setup.

The system also features an option for selecting CCTT facilities in advance of training. This is illustrated below in Figures 15 and 16 via *Site Selection* within the **Welcome Screen** main menu.

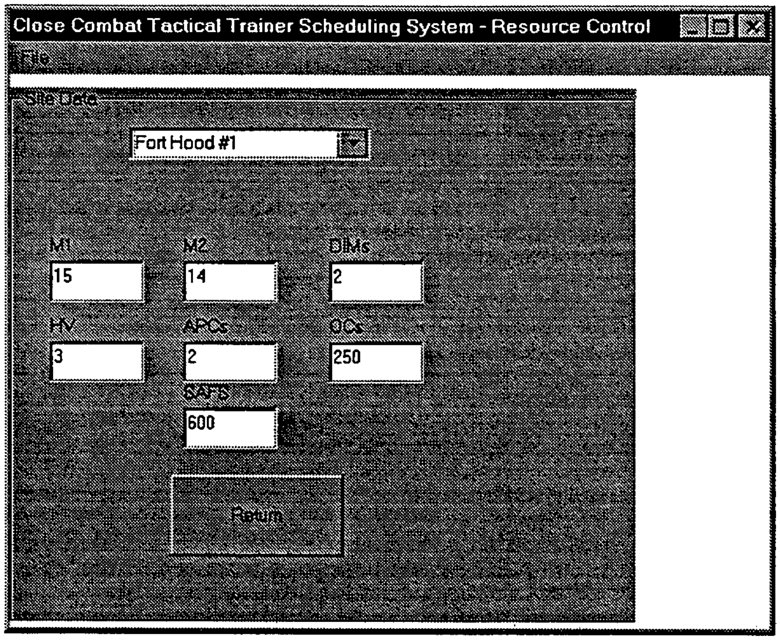

Figure 15. CCTT Site Selection.

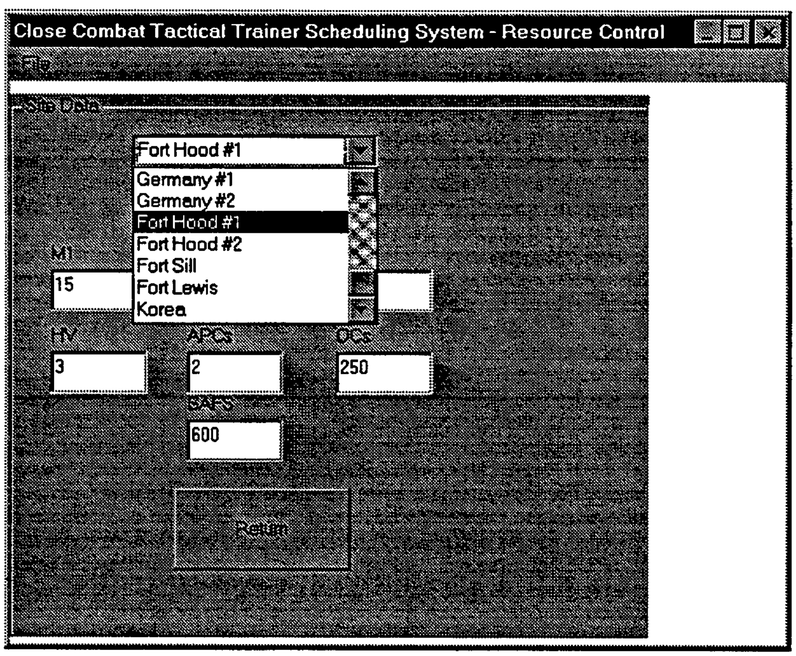

Figure 16. CCTT Site Change.

CCTT Training Site and training scenario data are also stored in the *Microsoft*® *Access*® database in the ***.ini*** file. Figures 17 and 18 demonstrate how the system user changes CCTT training site information that is stored in the Access database.

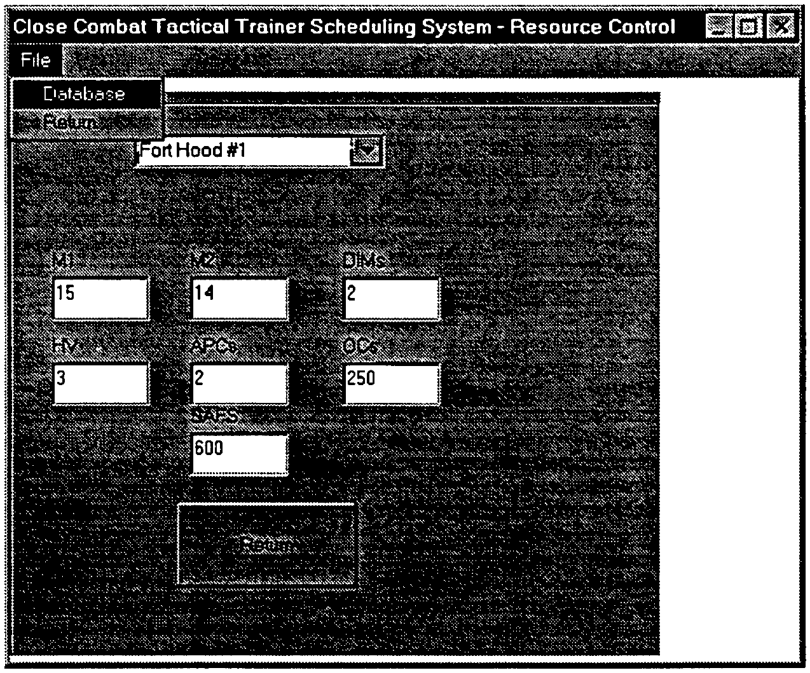

Figure 17. Step 1: CCTT Site Database Selection.

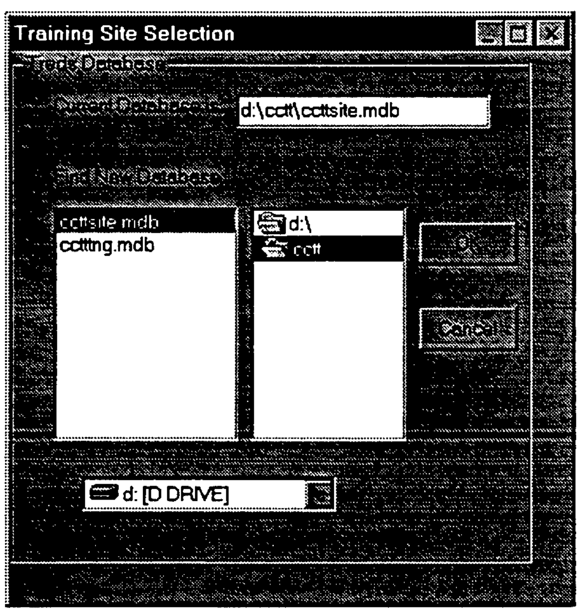

Figure 18. Step 2: CCTT Site Database Change.

Once the CCTT training system and training unit information have been initialized within the expert system, the next step is selecting training scenarios (events) for scheduling. Here, we assume the Battalion Commander or Battalion Operations Officer (i.e., S-3) decided to schedule units as follows: all platoons conduct a movement to contact; following this, all company teams conduct an attack; conclude with the battalion task force conducting a defend scenario. Figures 19 through 23 depict the screens for scheduling these events in the order specified.

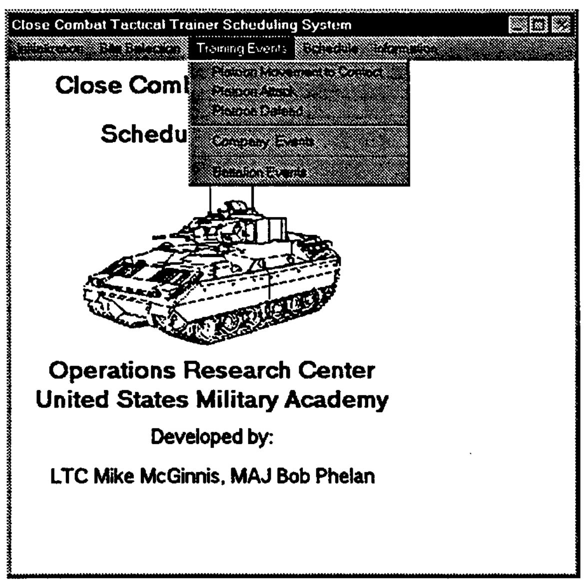

Figure 19. Select Training Events.

Clicking entries from the *Training Events* menu brings up the corresponding event selection screen. The user must then decide which events to schedule for training and the training context for the units undergoing training. Next, the training event lists are sorted. This is done first by training proficiency from *Untrained* to *Trained*, then, within a training proficiency category, by the user-specified priorities entered previously.

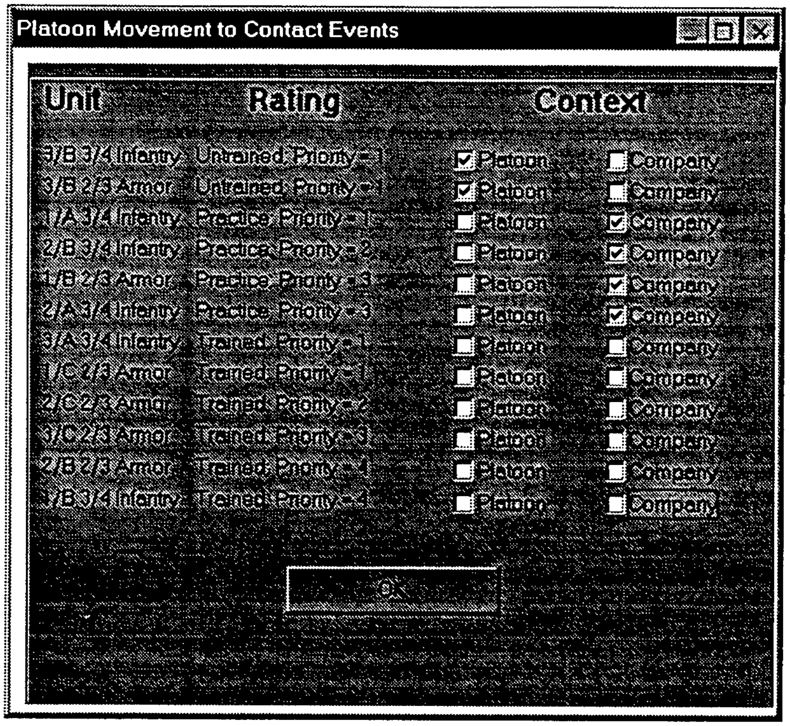

Figure 20. Select Platoon Movement to Contact Events.

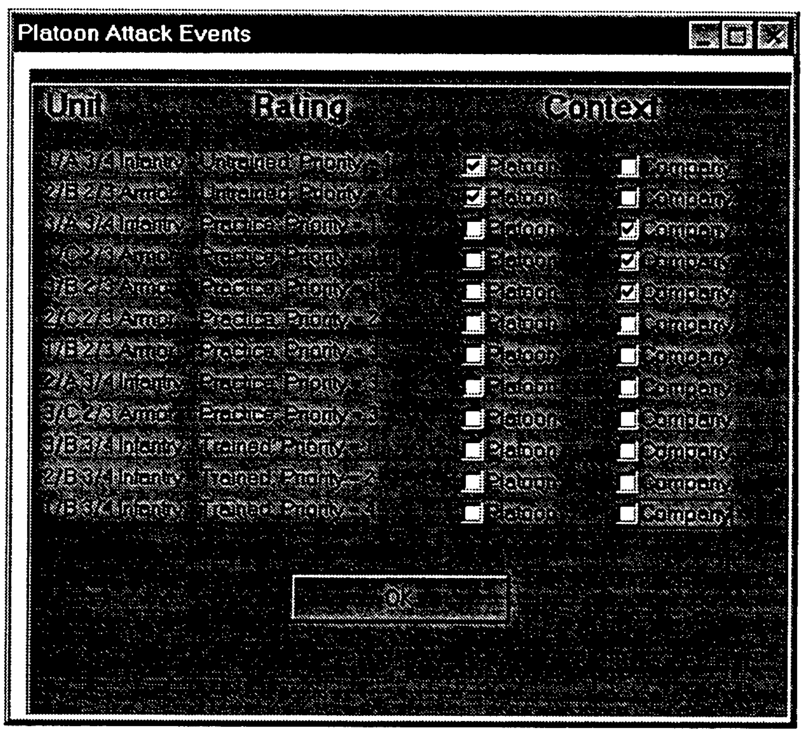

Figure 21. Select Platoon Attack Events.

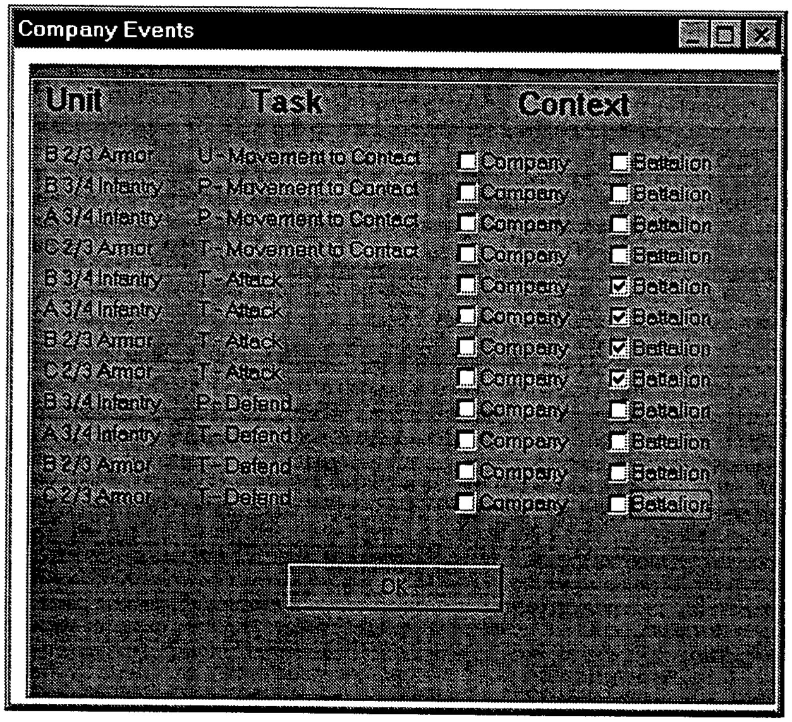

Figure 22. Select Company Events.

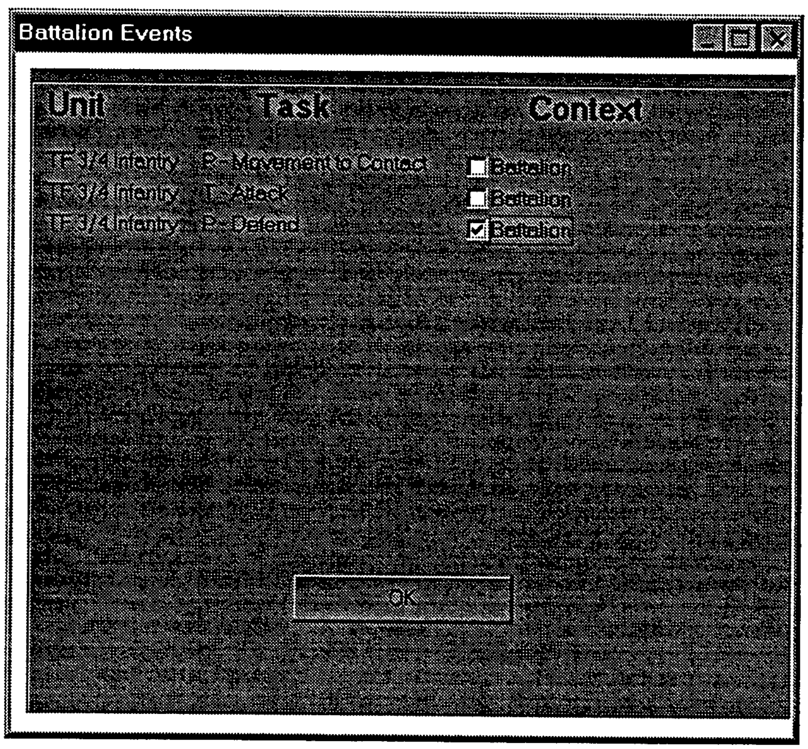

Figure 23. Select Battalion Events.

Once the system user selects all training scenarios (events), the scheduling program is executed by clicking on the *Schedule* option from the **Welcome Screen**'s main menu. The fully automated scheduling program automatically schedules training scenarios and RED SAF according to the pseudo-code of Appendix A. Scheduling results are presented to the user as shown below in Figure 24. The results include the training units conducting training in each hour, the scenarios to be conducted, and the level of resources (RED SAF) utilized. The training proficiency of each unit scheduled is shown color coded green (trained), amber (needs practice), and red (untrained).

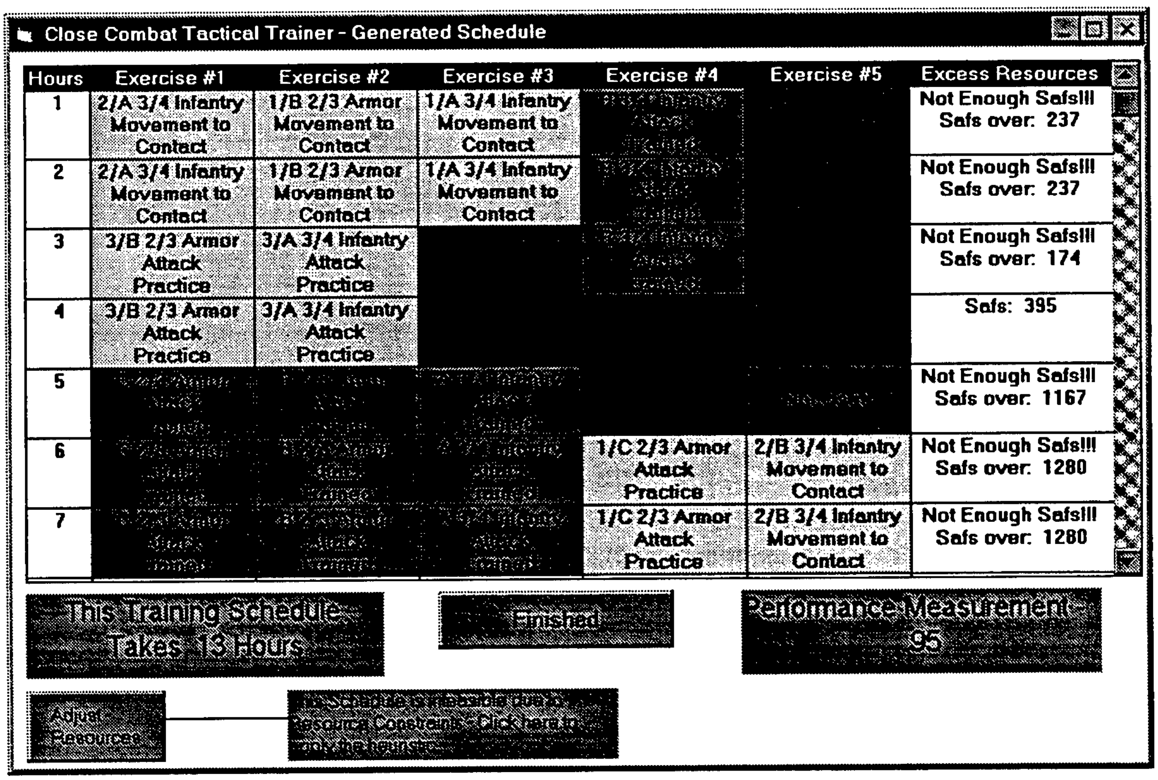

Figure 24. Schedule - First Seven Hours.

Additional scheduling information can be obtained by drilling down on any scheduled scenario. Figure 25 below illustrates the type of information provided to the system user by clicking on any of the events.

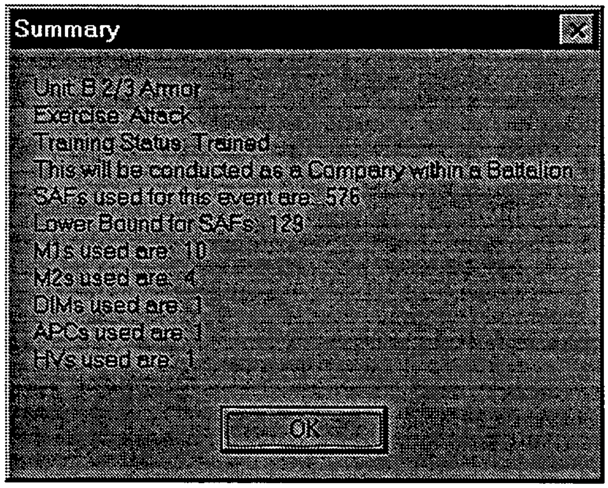

Figure 25. Event Detail.

Figure 29 reveals the rest of the training schedule that may be viewed by scrolling down the planning horizon

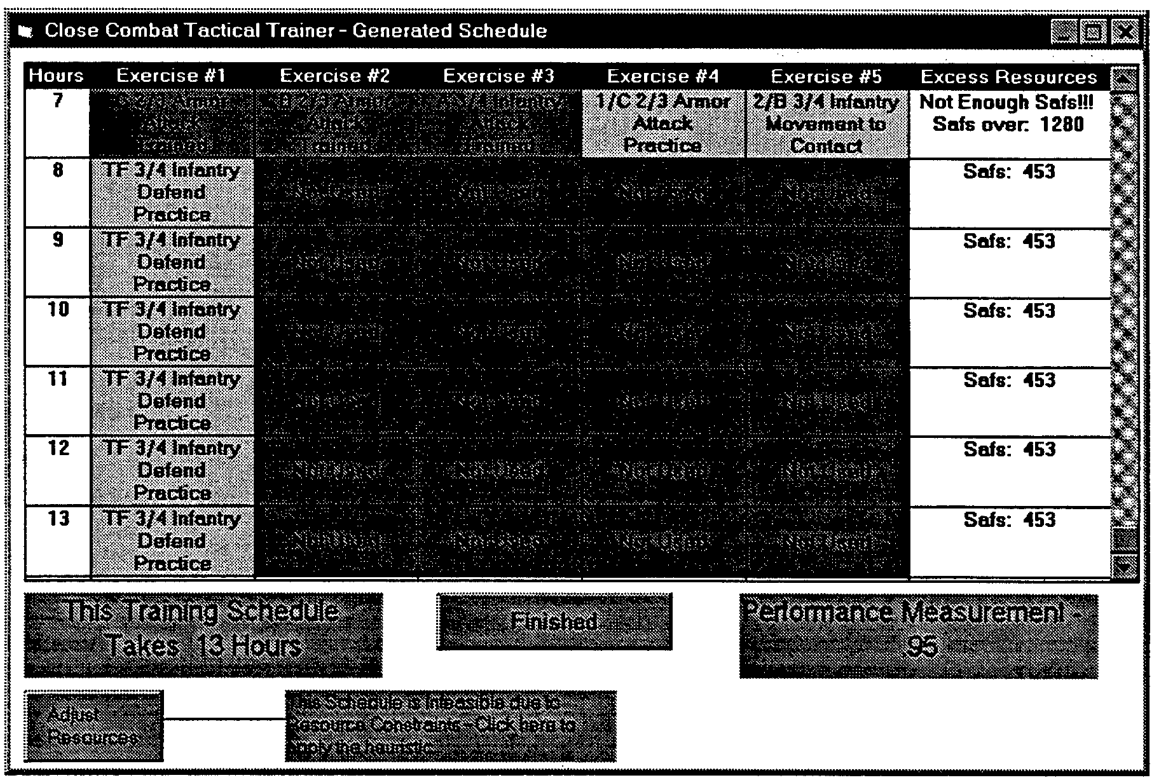

Figure 26. Remainder of Schedule

The last scheduling step invokes the *Heuristic Policy Improvement Procedure* that methodically and systematically improves the training resource schedule until no further improvements to the objective function are possible. This leads to the "best" training resource schedule obtainable with the heuristic scheduling program used here. The revised schedule is depicted in Figure 27.

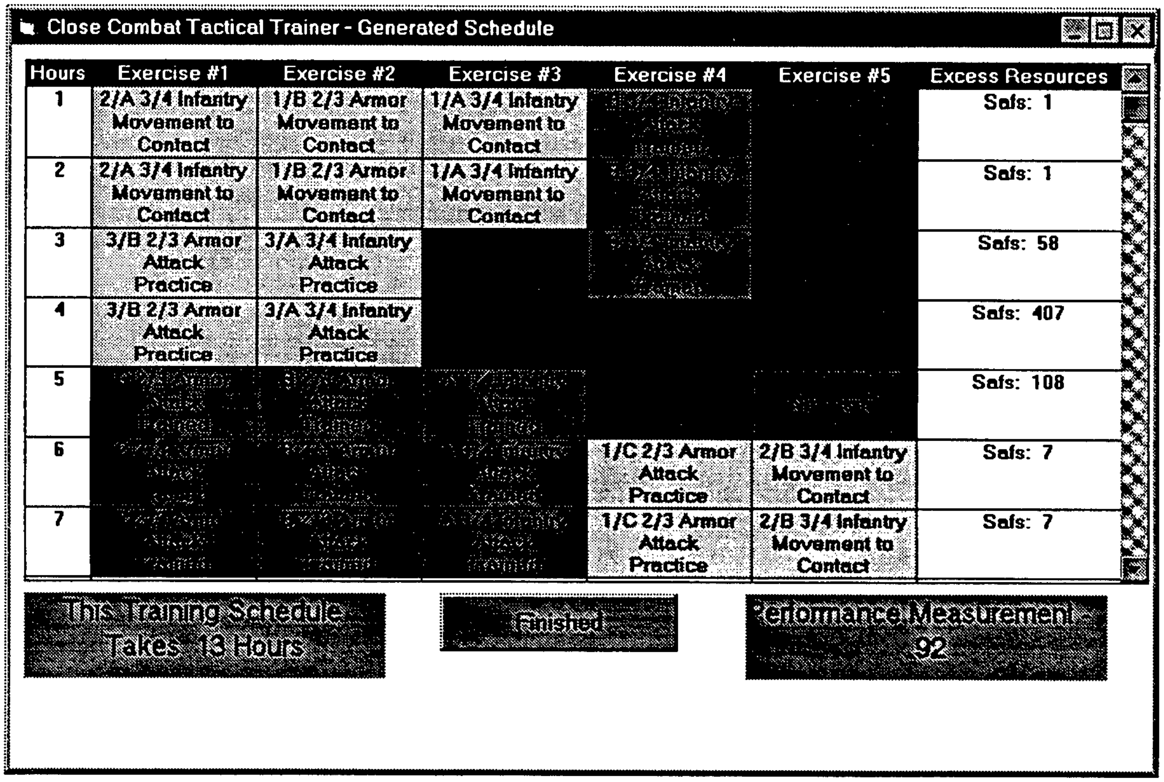

Figure 27. Training Resource Schedule following Heuristic Policy Improvement.

Once again, double clicking on any training event shows the adjusted SAF resources used for each training scenario (see Figure 28).

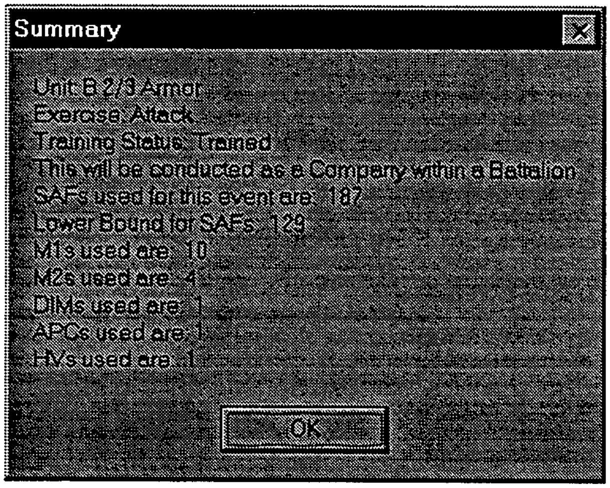

Figure 28. Adjusted Resources.

When the system user finishes a scheduling session, the user exits the expert system by simply selecting *Exit* from the *Initialization* option on the main menu.

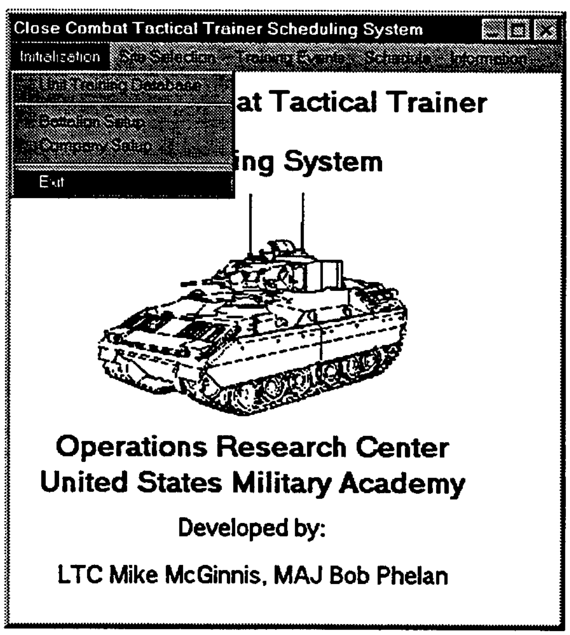

Figure 29. Exiting the Program.

APPENDIX D: CCTT HYBRID EXPERT SCHEDULING SYSTEM SOURCE CODE.

```
Attribute VB_Name = "Main"
Option Explicit
Public WrongTable As Boolean
Public WrongSiteTable As Boolean
Public Tng_File As String
Public Site_File As String
Public Save_Tab_Setting As Integer
Public Site_Save As String
Public M1, M2, Dims, Apc, Hv, Safs, Ocs As Integer
Public Schedule(306, 3) As Integer
Public Hours As Integer
Public Counter(5) As Integer
Public Unit_Str(17) As String
Public Unit_Assets(17)

Public Sub Update_Tabs()
Dim K As Integer
Save_Tab_Setting = MAIN_Screen.CO_DESG.Tab
K = 0
Do While K < 16
MAIN_Screen.CO_DESG.Tab = K
MAIN_Screen.CO_DESG.Caption = MAIN_Screen.UIC(K)
K = K + 1
Loop
MAIN_Screen.CO_DESG.Tab = Save_Tab_Setting
End Sub

Attribute VB_Name = "MAIN_Screen"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Public Data_Drive, Data_Dir, Data_File, SiteFile As String
Dim I, J, K, L, M, N, Pri As Integer
Public Rating As String
Dim Resultvalue As Long

Private Sub Battalion_setup_Click(Index As Integer)
MAIN_Screen.Caption = "Close Combat Tactical Trainer - Battalion/ Task
Force Setup"
' frm_Day.visible = False
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = True
End Sub

Private Sub cmd_Bn_Fin_Click()
MAIN_Screen.Caption = "Close Combat Tactical Trainer Scheduling System"

frm_Bnsetup.visible = False
Update_Tasks
End Sub

Private Sub CO_Entire_Click(Index As Integer)
I = 0
Do While I < 4

If CO_Entire(I).Value = True Then
    Plt_Entire(3 * I + 0).Enabled = True
    Plt_Entire(3 * I + 1).Enabled = True
    Plt_Entire(3 * I + 2).Enabled = True
```

```
    Plt_Minus(3 * I + 0).Enabled = True
    Plt_Minus(3 * I + 1).Enabled = True
    Plt_Minus(3 * I + 2).Enabled = True
    Plt_None(3 * I + 0).Enabled = False
    Plt_None(3 * I + 1).Enabled = False
    Plt_None(3 * I + 2).Enabled = False
    Plt_Entire(3 * I + 0) = True
    Plt_Entire(3 * I + 1) = True
    Plt_Entire(3 * I + 2) = True
End If
I = I + 1
Loop

End Sub

Private Sub CO_Not_Available_Click(Index As Integer)
I = 0
Do While I < 4
If CO_Not_Available(I).Value = True Then
    Plt_Entire(3 * I + 0).Enabled = False
    Plt_Entire(3 * I + 1).Enabled = False
    Plt_Entire(3 * I + 2).Enabled = False
    Plt_Minus(3 * I + 0).Enabled = False
    Plt_Minus(3 * I + 1).Enabled = False
    Plt_Minus(3 * I + 2).Enabled = False
    Plt_None(3 * I + 0).Enabled = True
    Plt_None(3 * I + 1).Enabled = True
    Plt_None(3 * I + 2).Enabled = True
    Plt_None(3 * I + 0).Value = True
    Plt_None(3 * I + 1).Value = True
    Plt_None(3 * I + 2).Value = True
End If
I = I + 1
Loop

End Sub

Private Sub CO_Plt_Only_Click(Index As Integer)
I = 0
Do While I < 4
If CO_Plt_Only(I).Value = True Then
    Plt_Entire(3 * I + 0).Enabled = True
    Plt_Entire(3 * I + 1).Enabled = True
    Plt_Entire(3 * I + 2).Enabled = True
    Plt_Minus(3 * I + 0).Enabled = False
    Plt_Minus(3 * I + 1).Enabled = False
    Plt_Minus(3 * I + 2).Enabled = False
    Plt_None(3 * I + 0).Enabled = True
    Plt_None(3 * I + 1).Enabled = True
    Plt_None(3 * I + 2).Enabled = True
If Plt_None(3 * I + 0).Value <> True Then
    Plt_Entire(3 * I + 0) = True
End If
If Plt_None(3 * I + 1).Value <> True Then
    Plt_Entire(3 * I + 1) = True
End If
If Plt_None(3 * I + 2).Value <> True Then
    Plt_Entire(3 * I + 2) = True
End If
 End If
 I = I + 1
```

```
Loop
End Sub

Private Sub CO_SETUP_Finished_Click()
MAIN_Screen.Caption = "Close Combat Tactical Trainer Scheduling System"
CO_SETUP_Finished.visible = False
CO_DESG.visible = False
Update_Tasks

End Sub

Private Sub CO_Staff_Click(Index As Integer)
I = 0
Do While I < 4
If CO_Staff(I).Value = True Then
    Plt_Entire(3 * I + 0).Enabled = True
    Plt_Entire(3 * I + 1).Enabled = True
    Plt_Entire(3 * I + 2).Enabled = True
    Plt_Minus(3 * I + 0).Enabled = True
    Plt_Minus(3 * I + 1).Enabled = True
    Plt_Minus(3 * I + 2).Enabled = True
    Plt_None(3 * I + 0).Enabled = True
    Plt_None(3 * I + 1).Enabled = True
    Plt_None(3 * I + 2).Enabled = True
If Plt_None(3 * I + 0).Value <> True Then
    Plt_Entire(3 * I + 0) = True
End If
If Plt_None(3 * I + 1).Value <> True Then
    Plt_Entire(3 * I + 1) = True
End If
If Plt_None(3 * I + 2).Value <> True Then
    Plt_Entire(3 * I + 2) = True
End If
End If
I = I + 1
Loop

End Sub

Private Sub Company_Setup_Click(Index As Integer)
MAIN_Screen.Caption = "Close Combat Tactical Trainer - Company/ Team
Setup"
' frm_Day.visible = False
Update_Tabs
frm_Bnsetup.visible = False
CO_DESG.visible = True
CO_SETUP_Finished.visible = True
End Sub

Private Sub data_init_Click()
' frm_Day.visible = False
Db_Change_Tng.Show
Db_Change_Tng.Database.visible = True
Db_Change_Tng.File1.visible = True
Db_Change_Tng.Tredsname.Text = Tng_File
End Sub

Private Sub Exit_Click()
SaveSetting "CCTT", "Startup", "File", Tng_File
SaveSetting "CCTT", "Startup", "SiteFile", Site_File
SaveSetting "CCTT", "Site", "Sitesave", Site_Save
```

```
dtaTasks.Recordset.Close

End

End Sub

Public Sub Update_Tasks()
' Rewrite the TPU Data file out
dtaTasks.Recordset.MoveFirst
M = 0
Do While M < 17
dtaTasks.Recordset.Edit
dtaTasks.Recordset.Fields(1) = UIC(M)
L = 2
Do While L < 5
      If unit_task((M * 9) + 3 * (L - 2) + 0) = True Then
        Rating = "T"
      End If
      If unit_task((M * 9) + 3 * (L - 2) + 1) = True Then
        Rating = "P"
      End If
      If unit_task((M * 9) + 3 * (L - 2) + 2) = True Then
        Rating = "U"
      End If
dtaTasks.Recordset.Fields(L) = Rating
L = L + 1
Loop
If opt_Arm(M).Value = True Then Rating = "Armor"
If opt_Mech(M).Value = True Then Rating = "Mech"
dtaTasks.Recordset.Fields(L) = Rating
dtaTasks.Recordset.Fields(L + 1) = txt_Priority(M).Text
M = M + 1
dtaTasks.Recordset.Update
dtaTasks.Recordset.MoveNext
Loop

End Sub

Private Sub Form_Load()
Dim I As Integer
Dim Load_Res As Boolean
Tng_File = GetSetting("CCTT", "Startup", "File")
Site_File = GetSetting("CCTT", "Startup", "SiteFile")
Site_Save = GetSetting("CCTT", "Site", "Sitesave")
I = 0
Hours = 12
I = 0
Do While I < 306
Schedule(I, 0) = 0
Schedule(I, 1) = -1
Schedule(I, 2) = 999
I = I + 1
Loop
Load_Res = True
On Error GoTo Error_mark
If Dir(Tng_File) <> "" And Tng_File <> "" Then
Initialize_Co
GoTo Bottom
Else
GoTo Error_mark
End If
```

```
Error_mark:
    Load_Res = False
msg = "Sorry - That is the Wrong Database as specified in the .ini file
- No Training Data" + Chr(13) + Chr(13) + "Please find the file in the
next screen."
style = vbOKOnly ' Define buttons.
title = "Database Error"

response = MsgBox(msg, style, title)
MAIN_Screen.visible = False
    Load Db_Change_Tng
    Db_Change_Tng.visible = True
'    Unload Me
    Db_Change_Tng.Show
Bottom:

N = 0
Do While N < 4
  CO_Entire(N).Value = True
  N = N + 1
Loop
Save_Tab_Setting = 0
CO_DESG.Tab = 0
Db_Change_Tng.Tredsname.Text = Tng_File

If Load_Res = True Then
Resource.visible = False
Load Resource
End If

End Sub

Private Sub mnu_Bn_Click()
Dim I As Integer
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = False
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
Load frm_Sel_event
frm_Sel_event.visible = False
frm_Sel_event.Battalion
frm_Sel_event.Caption = "Battalion Events"
frm_Sel_event.visible = True
frm_Sel_event.Label2.Caption = "Unit                Task
Context"
frm_Sel_event.Show
End Sub

Private Sub mnu_Co_Click()
Dim I As Integer
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = False
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
```

```
Loop
Load frm_Sel_event
frm_Sel_event.visible = False
frm_Sel_event.Company
frm_Sel_event.Caption = "Company Events"
frm_Sel_event.visible = True
frm_Sel_event.Label2.Caption = "Unit                Task
Context"
frm_Sel_event.Show

End Sub

Private Sub mnu_Info_Click()
msg = "For more information - Please contact " + Chr(13) + "The
Operations Research Center " + Chr(13) + "United States Military
Academy" + Chr(13) + Chr(13) + "LTC Mike McGinnis:" + Chr(13) +
"(DSN)688-5529" + Chr(13) + "(COMM) 914-938-5529" + Chr(13) +
"fm0768@exmail.usma.edu" + Chr(13) + Chr(13) + "MAJ Bob Phelan" +
Chr(13) + "(DSN)688-5941" + Chr(13) + "(COMM) 914-938-5941" + Chr(13) +
"fr0161@exmail.usma.edu" + Chr(13) + Chr(13) +
"http://www.orcen.usma.edu/"

style = vbOKOnly ' Define buttons.
title = "Operations Research Center Information"

response = MsgBox(msg, style, title)

End Sub

Private Sub mnu_Plt_Pr_Click()
Dim I As Integer
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = False
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.visible = False
frm_Sel_event.Practice
frm_Sel_event.Caption = "Platoon Attack Events"
frm_Sel_event.visible = True
frm_Sel_event.Label2.Caption = "Unit                Rating
Context"

frm_Sel_event.Show
End Sub

Private Sub mnu_Plt_tr_Click()
Dim I As Integer
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = False
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.visible = False
frm_Sel_event.Trained
```

```
frm_Sel_event.Caption = "Platoon Defend Events"
frm_Sel_event.visible = True
frm_Sel_event.Label2.Caption = "Unit                    Rating
Context"

frm_Sel_event.Show
End Sub

Private Sub mnu_PltU_Click()
Dim I As Integer
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = False
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.visible = False
frm_Sel_event.Untrained
frm_Sel_event.Caption = "Platoon Movement to Contact Events"
frm_Sel_event.visible = True
frm_Sel_event.Label2.Caption = "Unit                    Rating
Context"

frm_Sel_event.Show
End Sub

Private Sub mnu_PSCHED_Click()
Dim I As Integer
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.Untrained
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.Practice
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.Trained
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.Company
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
frm_Sel_event.Battalion
Load frm_Sch_List
```

```
End Sub

Private Sub opt_Arm_Click(Index As Integer)
Dim I, J, K, Count_Arm, Count_Co As Integer
Count_Arm = 0
Count_Co = 0
I = 1
J = 0
If opt_Arm(Index).Value = True Then
  Unit_Str(Index) = "Armor"
Else
  Unit_Str(Index) = "Mech"
End If

Do While I < 17
'If opt_Arm(I).Value = True Then Count_Arm = Count_Arm + 1
  I = I + 1
  If I = 4 Or I = 8 Or I = 12 Or I = 16 Then
    K = 1
    Do While K < 4
      If opt_Arm(I - 4 + K).Value = True Then Count_Arm = Count_Arm + 1
      K = K + 1
    Loop
  If Count_Arm = 0 Then
    If opt_Arm(I - 4).Value = False Then
      Lbl_TO(J).Caption = "Pure Mech Company"
    Else
      Lbl_TO(J).Caption = "Three Mech Platoons with an Armor Co HQ"
    End If
    Unit_Str(I - 4) = "Pure Mech"
  End If
  If Count_Arm = 1 Then
    If opt_Arm(I - 4).Value = False Then
      Lbl_TO(J).Caption = "Mech Heavy Team"
    Else
      Lbl_TO(J).Caption = "Mech Heavy Team with an Armor Co HQ"
    End If
    Unit_Str(I - 4) = "Heavy Mech"
  End If
  If Count_Arm = 2 Then
    If opt_Arm(I - 4).Value = True Then
      Lbl_TO(J).Caption = "Armor Heavy Team"
    Else
      Lbl_TO(J).Caption = "Armor Heavy Team with an Mech Co HQ"
    End If
    Unit_Str(I - 4) = "Heavy Armor"
  End If
  If Count_Arm = 3 Then
    If opt_Arm(I - 4).Value = True Then
      Lbl_TO(J).Caption = "Pure Armor Company"
    Else
      Lbl_TO(J).Caption = "Three Armor Platoons with a Mech Co HQ"
    End If
    Unit_Str(I - 4) = "Pure Armor"

  End If
  J = J + 1
  I = I + 1
  Count_Arm = 0
End If
Loop
```

```
J = 0
Do While J < 16
  If opt_Arm(J).Value = True Then Count_Co = Count_Co + 1
  J = J + 4
Loop

  If Count_Co = 0 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Mechanized Infantry Battalion. (4 Mech)"
    Unit_Str(16) = "Pure Mech"
  End If
  If Count_Co = 0 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Four Mechanized Infantry Companies with an Armor
Battalion HQ."
    Unit_Str(16) = "Pure Mech"
  End If
  If Count_Co = 4 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Armor Battalion. (4 Armor)"
    Unit_Str(16) = "Pure Armor"
  End If
  If Count_Co = 4 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Four Armor Companies with a Mechanized Infantry
Battalion HQ."
    Unit_Str(16) = "Pure Armor"
  End If
  If Count_Co = 1 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Heavy Mechanized Infantry Task Force (3 Mech, 1 Armor)
with an Armor Battalion HQ"
    Unit_Str(16) = "Heavy Mech"
  End If
  If Count_Co = 1 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Heavy Mechanized Infantry Task Force. (3 Mech, 1
Armor)"
    Unit_Str(16) = "Heavy Mech"
  End If
  If Count_Co = 2 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Balanced Armor Task Force. (2 Armor 2 Mech)"
    Unit_Str(16) = "Balanced Armor"
  End If
  If Count_Co = 2 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Balanced Mechanized Infantry Task Force. (2 Mech, 2
Armor)"
    Unit_Str(16) = "Balanced Mech"
  End If
  If Count_Co = 3 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Heavy Armor Task Force. (3 Armor, 1 Mech)"
    Unit_Str(16) = "Heavy Armor"
  End If
  If Count_Co = 3 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Heavy Armor Task Force (3 Armor, 1 Mech) with a
Mechanized Infantry Battalion HQ."
    Unit_Str(16) = "Heavy Armor"
  End If
End Sub

Private Sub opt_Bnall_Click()
Dim I As Integer
I = 0
Do While I < 4
  CO_Entire(I).Value = True
  CO_Staff(I).Enabled = False
```

```
    CO_Not_Available(I).Enabled = False
    CO_Plt_Only(I).Enabled = False

  I = I + 1
  Loop
  I = 0
  Do While I < 8
    Plt_Minus(I).Enabled = False
    I = I + 1

  Loop
  mnu_Bn.Enabled = True
  End Sub

  Private Sub opt_bnno_Click()
  Dim I As Integer
  I = 0
  Do While I < 4
    CO_Staff(I).Enabled = True
    CO_Not_Available(I).Enabled = True
    CO_Plt_Only(I).Enabled = True

  I = I + 1
  Loop
  I = 0
  Do While I < 8
    Plt_Minus(I).Enabled = True
  I = I + 1
  Loop
  mnu_Bn.Enabled = False
  End Sub

  Private Sub opt_Bnsome_Click()
  Dim I As Integer
  I = 0
  Do While I < 4
    CO_Staff(I).Enabled = True
    CO_Not_Available(I).Enabled = True
    CO_Plt_Only(I).Enabled = True

  I = I + 1
  Loop
  I = 0
  Do While I < 8
    Plt_Minus(I).Enabled = True
  I = I + 1
  Loop
  mnu_Bn.Enabled = True
  End Sub

  Private Sub opt_Mech_Click(Index As Integer)
  Dim I, J, K, Count_Arm, Count_Co As Integer
  Count_Arm = 0
  Count_Co = 0
  I = 1
  J = 0
  If opt_Arm(Index).Value = True Then
    Unit_Str(Index) = "Armor"
  Else
    Unit_Str(Index) = "Mech"
  End If
```

```
Do While I < 17
'If opt_Arm(I).Value = True Then Count_Arm = Count_Arm + 1
  I = I + 1
  If I = 4 Or I = 8 Or I = 12 Or I = 16 Then
    K = 1
    Do While K < 4
      If opt_Arm(I - 4 + K).Value = True Then Count_Arm = Count_Arm + 1
      K = K + 1
    Loop
  If Count_Arm = 0 Then
    If opt_Arm(I - 4).Value = False Then
      Lbl_TO(J).Caption = "Pure Mech Company"
    Else
      Lbl_TO(J).Caption = "Three Mech Platoons with an Armor Co HQ"
    End If
    Unit_Str(I - 4) = "Pure Mech"
  End If
  If Count_Arm = 1 Then
    If opt_Arm(I - 4).Value = False Then
      Lbl_TO(J).Caption = "Mech Heavy Team"
    Else
      Lbl_TO(J).Caption = "Mech Heavy Team with an Armor Co HQ"
    End If
    Unit_Str(I - 4) = "Heavy Mech"
  End If
  If Count_Arm = 2 Then
    If opt_Arm(I - 4).Value = True Then
      Lbl_TO(J).Caption = "Armor Heavy Team"
    Else
      Lbl_TO(J).Caption = "Armor Heavy Team with an Mech Co HQ"
    End If
    Unit_Str(I - 4) = "Heavy Armor"
  End If
  If Count_Arm = 3 Then
    If opt_Arm(I - 4).Value = True Then
      Lbl_TO(J).Caption = "Pure Armor Company"
    Else
      Lbl_TO(J).Caption = "Three Armor Platoons with a Mech Co HQ"
    End If
    Unit_Str(I - 4) = "Pure Armor"

  End If
  J = J + 1
  I = I + 1
  Count_Arm = 0
End If
Loop

J = 0
Do While J < 16
  If opt_Arm(J).Value = True Then Count_Co = Count_Co + 1
  J = J + 4
Loop

  If Count_Co = 0 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Mechanized Infantry Battalion. (4 Mech)"
    Unit_Str(16) = "Pure Mech"
  End If
  If Count_Co = 0 And opt_Arm(16).Value = True Then
```

```
    Lbl_TO(4) = "Four Mechanized Infantry Companies with an Armor
Battalion HQ."
    Unit_Str(16) = "Pure Mech"
  End If
  If Count_Co = 4 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Armor Battalion. (4 Armor)"
    Unit_Str(16) = "Pure Armor"
  End If
  If Count_Co = 4 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Four Armor Companies with a Mechanized Infantry
Battalion HQ."
    Unit_Str(16) = "Pure Armor"
  End If
  If Count_Co = 1 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Heavy Mechanized Infantry Task Force (3 Mech, 1 Armor)
with an Armor Battalion HQ"
    Unit_Str(16) = "Heavy Mech"
  End If
  If Count_Co = 1 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Heavy Mechanized Infantry Task Force. (3 Mech, 1
Armor)"
    Unit_Str(16) = "Heavy Mech"
  End If
  If Count_Co = 2 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Balanced Armor Task Force. (2 Armor 2 Mech)"
    Unit_Str(16) = "Balanced Armor"
  End If
  If Count_Co = 2 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Balanced Mechanized Infantry Task Force. (2 Mech, 2
Armor)"
    Unit_Str(16) = "Balanced Mech"
  End If
  If Count_Co = 3 And opt_Arm(16).Value = True Then
    Lbl_TO(4) = "Heavy Armor Task Force. (3 Armor, 1 Mech)"
    Unit_Str(16) = "Heavy Armor"
  End If
  If Count_Co = 3 And opt_Arm(16).Value = False Then
    Lbl_TO(4) = "Heavy Armor Task Force (3 Armor, 1 Mech) with a
Mechanized Infantry Battalion HQ."
    Unit_Str(16) = "Heavy Armor"
  End If
End Sub

Private Sub Site_Init_Click()
CO_DESG.visible = False
CO_SETUP_Finished.visible = False
frm_Bnsetup.visible = False
Tng_Site_Selection.Tredsname = Site_File
Resource.WindowState = 2
Resource.Show
MAIN_Screen.visible = False
End Sub

Private Sub txt_Priority_Change(Index As Integer)
Dim J As Integer
Dim Str As String
J = CO_DESG.Tab
Str = txt_Priority(J).Text
If Str Like "" Or Str Like "1" Or Str Like "2" Or Str Like "3" Or Str
Like "4" Or Str Like "5" Then
Else
```

```
txt_Priority(J).Text = "5"
MsgBox "Priority must be between 1 and 5"
End If

End Sub

Private Sub txt_Priority_LostFocus(Index As Integer)
Dim J As Integer
Dim Str As String
J = CO_DESG.Tab
Str = txt_Priority(J).Text
If Str Like "" Then txt_Priority(J).Text = "5"

End Sub

Private Sub UIC_Change(Index As Integer)
Dim J As Integer
J = CO_DESG.Tab
CO_DESG.Caption = UIC(J)
End Sub

Public Sub Initialize_Co()
dtaTasks.DatabaseName = Tng_File
dtaTasks.RecordSource = "Task"
Db_Change_Tng.Tredsname.Text = Tng_File
On Error GoTo Errorhandler
dtaTasks.Refresh
dtaTasks.Recordset.MoveFirst
M = 0
Do While M < 17
  UIC(M).Text = dtaTasks.Recordset.Fields(1)
  L = 2
  Do While L < 5
    Rating = dtaTasks.Recordset.Fields(L)
    If Rating = "T" Then
      unit_task(M * 9 + (L - 2) * 3 + 0).Value = True
    End If
    If Rating = "P" Then
      unit_task(M * 9 + (L - 2) * 3 + 1).Value = True
    End If
    If Rating = "U" Then
      unit_task(M * 9 + (L - 2) * 3 + 2).Value = True
    End If
   L = L + 1
  Loop

'
' Grab Task Organization Here - L is 5
'
Rating = dtaTasks.Recordset.Fields(L)
If Rating = "Armor" Then opt_Arm(M).Value = True
If Rating = "Mech" Then opt_Mech(M).Value = True
txt_Priority(M) = dtaTasks.Recordset.Fields(L + 1)
M = M + 1

dtaTasks.Recordset.MoveNext
Loop
MAIN_Screen.Show
GoTo Bottom

Errorhandler:
```

```
    WrongTable = False
    Db_Change_Tng.Show
    Db_Change_Tng.Database.visible = True
    Db_Change_Tng.File1.visible = True
    Db_Change_Tng.Tredsname.Text = Tng_File
Bottom:
CO_DESG.Tab = 0
Opt_Bnall.Value = True
End Sub

Attribute VB_Name = "Db_Change_Tng"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Public Data_Drive, Data_Dir, Data_File As String
Dim I, J, K, L, M, N As Integer
Public Rating As String
Dim Resultvalue As Long
Dim Flag As Boolean

Private Sub data_Exit_Click()
If Flag = False Then
End
Else
Unload Me
End If
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
File1.Pattern = "*.mdb"
End Sub

Private Sub Drive_Cmd_Click()
File1.Pattern = "*.mdb"
ChDrive Drive1.Drive
ChDir Dir1.Path
Data_Drive = Drive1.Drive
Data_Dir = Dir1.Path
Data_File = File1.filename
Tng_File = Dir1.Path + "\" + File1.filename
MAIN_Screen.dtaTasks.DatabaseName = Tng_File
Db_Change_Tng.Tredsname.Text = Tng_File
On Error GoTo Errorhandler
MAIN_Screen.Initialize_Co
' If WrongTable = False Then GoTo Errorhandler
Db_Change_Tng.visible = False
MAIN_Screen.visible = True
Flag = True
Unload Me
GoTo Bottom
Errorhandler:
msg = "Sorry - That is the Wrong Database"
Style = vbOKOnly ' Define buttons.
Title = "Database Error"

response = MsgBox(msg, Style, Title)

    MsgBox response
    Flag = False
    Db_Change_Tng.Show
    Db_Change_Tng.Database.visible = True
```

```
    Db_Change_Tng.File1.visible = True
    WrongTable = True
Bottom:
Resource.visible = False
Load Resource
Err.Number = 0

End Sub

Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub

Private Sub File1_DblClick()
File1.Pattern = "*.mdb"
ChDrive Drive1.Drive
ChDir Dir1.Path
Data_Drive = Drive1.Drive
Data_Dir = Dir1.Path
Data_File = File1.filename
Tng_File = Dir1.Path + "\" + File1.filename
MAIN_Screen.dtaTasks.DatabaseName = Tng_File
Db_Change_Tng.Tredsname.Text = Tng_File
On Error GoTo Errorhandler
MAIN_Screen.Initialize_Co
If WrongTable = False Then GoTo Errorhandler
Db_Change_Tng.visible = False
MAIN_Screen.visible = True
Flag = True
Unload Me
GoTo Bottom
Errorhandler:

msg = "Sorry - That is the Wrong Database"
Style = vbOKOnly ' Define buttons.
Title = "Database Error"

response = MsgBox(msg, Style, Title)

    MsgBox response
    Flag = False
    Db_Change_Tng.Show
    Db_Change_Tng.Database.visible = True
    Db_Change_Tng.File1.visible = True
Bottom:
Resource.visible = False
Load Resource
Err.Number = 0


End Sub

Private Sub Form_Load()
File1.Pattern = "*.mdb"
Flag = True
Tredsname.Text = Tng_File
'Visible = True

End Sub

Attribute VB_Name = "frm_Sch_List"
```

```
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Dim Sch_Str_Arr(7, 61)
Dim Unit_Str_Arr(7, 61, 8)
Dim Exercise, Proficiency, Echelon, Unit_Type As String
Dim Sch_Str, Sch_Str1
Dim Saf_Array(7, 61, 8) As Integer
Dim Saf_number, Num_Rows_Tbl As Integer
Dim Perf_Meas(60) As Single
Dim Resource_flag As Boolean

Private Sub cmd_Adjust_Click()
Adjust_Res
  cmd_Adjust.visible = False
  res_label.visible = False
  line1.visible = False
End Sub

Private Sub Command1_Click()
dta_Blue.Recordset.Close
dta_scen.Recordset.Close

Unload Me
End Sub

Public Sub Form_Load()
Dim Units(99, 12) As Integer

Dim Num_Events, Saf_Total, Internal_Counter, Internal_counter2 As
Integer
Dim I, J, K, L, M, Num_Rows, Time, Schedule_Value As Integer
Dim Hour(300, 12) As Long
Dim Platoons(60) As Integer
Dim Company(60) As Integer
Dim Battalion(60) As Integer
Dim Saf_Row(60) As Integer
Dim Event, Rnge As String
Dim Unitflag As Boolean

Dim Temp_PM As Single
MsgBox "Initializing"
I = 0
Do While I < 7
  K = 0
    Do While K < 61
      L = 0
        Do While L < 8
          Saf_Array(I, K, L) = 0
          Unit_Str_Arr(I, K, L) = 0
          L = L + 1
        Loop
      K = K + 1
    Loop
  I = I + 1
Loop

K = 0
L = 0
Time = 0
I = 0
dta_scen.DatabaseName = Site_File
```

```
dta_scen.RecordSource = "Scenarios"
dta_scen.Refresh
dta_Blue.DatabaseName = Site_File
dta_Blue.RecordSource = "Blue"
dta_Blue.Refresh
Hours = 60

I = 0
Do While I < 100
  Units(I, 0) = -1
  Units(I, 1) = -1
  Units(I, 2) = -1
  Units(I, 3) = -1
  Units(I, 4) = 0 ' Safs
  Units(I, 5) = 0 ' M1
  Units(I, 6) = 0 ' M2
  Units(I, 7) = 0 ' Dim
  Units(I, 8) = 0 ' APC
  Units(I, 9) = 0 ' HV
  Units(I, 10) = 0 ' OCs
  Units(I, 11) = 0 'SAF lower Bound
  I = I + 1
Loop
I = 0
J = 0
Do While I < 300
    Hour(I, 0) = -2
    Hour(I, 1) = -2
    Hour(I, 2) = -2
    Hour(I, 3) = -2
    Hour(I, 4) = 0 ' safs
    Hour(I, 5) = 0 ' M1
    Hour(I, 6) = 0 ' M2
    Hour(I, 7) = 0 ' Dim
    Hour(I, 8) = 0 ' APC
    Hour(I, 9) = 0 ' HV
    Hour(I, 10) = 0 ' OCs
    Hour(I, 11) = 0 ' Saf Lower Bound
  I = I + 1
  Loop
Do While I < 60
  Platoons(I) = 0
  Company(I) = 0
  Battalion(I) = 0
  Sch_Str_Arr(0, I) = "0"
  Sch_Str_Arr(1, I) = "1"
  Sch_Str_Arr(2, I) = "2"
  Sch_Str_Arr(3, I) = "3"
  Sch_Str_Arr(4, I) = "4"
  Sch_Str_Arr(5, I) = "5"
  Saf_Row(I) = 0
  Perf_Meas(I) = 0
  I = I + 1
Loop
MsgBox "Building Event List and cross references"
I = 0
Do While I < 306
  J = 0
  M = 0
  Do While J < 306
    M = J
```

```
    If J >= 153 Then M = J - 153
    If Schedule(J, 2) = I And Schedule(J, 0) = 1 And
MAIN_Screen.unit_task(M).Value = True Then
      L = Int(J / 9)
      If L > 16 Then
        L = L - 17
        M = (J - 153) Mod 9
      Else
        M = J Mod 9
      End If
      Units(K, 0) = L
      Units(K, 1) = 0
      Units(K, 3) = J
      If L = 0 Or L = 4 Or L = 8 Or L = 12 Then Units(K, 1) = 1
      If L = 16 Then
        If M = 0 Or M = 1 Or M = 2 Then Units(K, 1) = 2
        If M = 3 Or M = 4 Or M = 5 Then Units(K, 1) = 3
        If M = 6 Or M = 7 Or M = 8 Then Units(K, 1) = 4
      End If
      Units(K, 4) = SAF_FCN(J, L)
      Units(K, 5) = Unit_FCN(J, L, 5)
      Units(K, 6) = Unit_FCN(J, L, 6)
      Units(K, 7) = Unit_FCN(J, L, 7)
      Units(K, 8) = Unit_FCN(J, L, 8)
      Units(K, 9) = Unit_FCN(J, L, 9)
      Units(K, 10) = Unit_FCN(J, L, 10)
      Units(K, 11) = SAF_LB(J, L)

      K = K + 1
      J = 999
    End If
  J = J + 1
  Loop
I = I + 1
Loop
MsgBox "Starting Algorithm"
'Step A
If K > 0 Then
Time = Hours - 1
'Step B
Do While Time > -1
'Step C
  Schedule_Value = 0
  Num_Events = Platoons(Time) + Company(Time) + Battalion(Time)
 'Step D
 'Step E
   'select a platoon event if any
         I = 98
         Schedule_Value = 0
         Do While I > -1
             Schedule_Value = 0
             If Platoons(Time) > 0 Then Schedule_Value = 2 ^
Platoons(Time) + Schedule_Value
             If Company(Time) > 0 Then Schedule_Value = 3 ^ Company(Time)
+ Schedule_Value
             If Battalion(Time) > 0 Then Schedule_Value = 32
 'step f
 'MTC
        If Schedule_Value = 0 And Time >= 2 And Units(I, 1) = 2 And
Units(I, 2) = -1 Then
              J = 0
```

```
            Do While J < 5
              If Hour(J + 5 * Time, 0) = -2 And Hour(J + 5 * Time, 1) =
-2 And Hour(J + 5 * Time, 2) = -2 Then
                Battalion(Time) = Battalion(Time) + 1
                Battalion(Time - 1) = Battalion(Time - 1) + 1
                Battalion(Time - 2) = Battalion(Time - 2) + 1
                'shut off unavailable blocks
                Hour(J + 5 * Time, 2) = Units(I, 0)
                Hour(J + 5 * (Time - 1), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 2), 2) = Units(I, 0)
                Internal_Counter = 3
                Do While Internal_Counter < 12
                  Hour(J + 5 * Time, Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 1), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 2), Internal_Counter) = Units(I,
Internal_Counter)
                  Internal_Counter = Internal_Counter + 1
                Loop
                Hour(J + 5 * Time, 0) = -1
                Hour(J + 5 * Time, 1) = -1
                Units(I, 2) = J + 5 * Time
                J = 5
              End If 'Hour
              J = J + 1
            Loop 'j
  'atk
          ElseIf Schedule_Value = 0 And Time >= 4 And Units(I, 1) = 3
And Units(I, 2) = -1 Then
            J = 0
            Do While J < 5
              If Hour(J + 5 * Time, 0) = -2 And Hour(J + 5 * Time, 1) =
-2 And Hour(J + 5 * Time, 2) = -2 Then
                Battalion(Time) = Battalion(Time) + 1
                Battalion(Time - 1) = Battalion(Time - 1) + 1
                Battalion(Time - 2) = Battalion(Time - 2) + 1
                Battalion(Time - 3) = Battalion(Time - 3) + 1
                Battalion(Time - 4) = Battalion(Time - 4) + 1
                'shut off unavailable blocks
                Hour(J + 5 * Time, 2) = Units(I, 0)
                Hour(J + 5 * (Time - 1), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 2), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 3), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 4), 2) = Units(I, 0)
                Internal_Counter = 3
                Do While Internal_Counter < 12
                  Hour(J + 5 * Time, Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 1), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 2), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 3), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 4), Internal_Counter) = Units(I,
Internal_Counter)
                  Internal_Counter = Internal_Counter + 1
                Loop
                Hour(J + 5 * Time, 0) = -1
                Hour(J + 5 * Time, 1) = -1
```

```
                Units(I, 2) = J + 5 * Time
                J = 5
              End If 'Hour
              J = J + 1
            Loop 'j
'def
          ElseIf Schedule_Value = 0 And Time >= 5 And Units(I, 1) = 4
And Units(I, 2) = -1 Then
            J = 0
            Do While J < 5
              If Hour(J + 5 * Time, 0) = -2 And Hour(J + 5 * Time, 1) =
-2 And Hour(J + 5 * Time, 2) = -2 Then
                Battalion(Time) = Battalion(Time) + 1
                Battalion(Time - 1) = Battalion(Time - 1) + 1
                Battalion(Time - 2) = Battalion(Time - 2) + 1
                Battalion(Time - 3) = Battalion(Time - 3) + 1
                Battalion(Time - 4) = Battalion(Time - 4) + 1
                Battalion(Time - 5) = Battalion(Time - 5) + 1
                'shut off unavailable blocks
                Hour(J + 5 * Time, 2) = Units(I, 0)
                Hour(J + 5 * (Time - 1), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 2), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 3), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 4), 2) = Units(I, 0)
                Hour(J + 5 * (Time - 5), 2) = Units(I, 0)
                Internal_Counter = 3
                Do While Internal_Counter < 12
                  Hour(J + 5 * Time, Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 1), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 2), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 3), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 4), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 5), Internal_Counter) = Units(I,
Internal_Counter)
                  Internal_Counter = Internal_Counter + 1
                Loop
                Hour(J + 5 * Time, 0) = -1
                Hour(J + 5 * Time, 1) = -1
                Units(I, 2) = J + 5 * Time
                J = 5
              End If 'Hour
              J = J + 1
            Loop 'j
 'step g
        ElseIf Schedule_Value <= 29 And Company(Time) < 3 And Time >= 2
And Units(I, 1) = 1 And Units(I, 2) = -1 Then
            J = 0
            Unitflag = True
            Do While J < 5
              If Hour(J + 5 * Time, 1) = Units(I, 0) Then Unitflag =
False
              If Units(I, 0) = 0 And (Hour(J + 5 * Time, 0) = 1 Or
Hour(J + 5 * Time, 0) = 2 Or Hour(J + 5 * Time, 0) = 3) Then Unitflag =
False
```

```
              If Units(I, 0) = 4 And (Hour(J + 5 * Time, 0) = 5 Or
Hour(J + 5 * Time, 0) = 6 Or Hour(J + 5 * Time, 0) = 7) Then Unitflag =
False
              If Units(I, 0) = 8 And (Hour(J + 5 * Time, 0) = 9 Or
Hour(J + 5 * Time, 0) = 10 Or Hour(J + 5 * Time, 0) = 11) Then Unitflag
= False
              If Units(I, 0) = 12 And (Hour(J + 5 * Time, 0) = 13 Or
Hour(J + 5 * Time, 0) = 14 Or Hour(J + 5 * Time, 0) = 15) Then Unitflag
= False
              J = J + 1
            Loop
            J = 0
            Do While J < 5
              If Units(I, 2) = -1 And Unitflag = True And Hour(J + 5 *
Time, 0) = -2 And Hour(J + 5 * Time, 1) = -2 And Hour(J + 5 * Time, 2) =
-2 Then ' And Units(I, 4) + Saf_Row(Time) < Safs Then
                Company(Time) = Company(Time) + 1
                Company(Time - 1) = Company(Time - 1) + 1
                Company(Time - 2) = Company(Time - 2) + 1
                'shut off unavailable blocks
                Hour(J + 5 * Time, 1) = Units(I, 0)
                Hour(J + 5 * (Time - 1), 1) = Units(I, 0)
                Hour(J + 5 * (Time - 2), 1) = Units(I, 0)
                Internal_Counter = 3
                Do While Internal_Counter < 12
                  Hour(J + 5 * Time, Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 1), Internal_Counter) = Units(I,
Internal_Counter)
                  Hour(J + 5 * (Time - 2), Internal_Counter) = Units(I,
Internal_Counter)
                  Internal_Counter = Internal_Counter + 1
                Loop

                Hour(J + 5 * Time, 0) = -1
                Hour(J + 5 * Time, 2) = -1
                Units(I, 2) = J + 5 * Time

                Saf_Row(Time) = Saf_Row(Time) + Units(I, 4)
                Saf_Row(Time - 1) = Saf_Row(Time - 1) + Units(I, 4)
                Saf_Row(Time - 2) = Saf_Row(Time - 2) + Units(I, 4)
                J = 5
              End If 'Hour
              J = J + 1
            Loop 'j
            ElseIf Schedule_Value <= 30 And Platoons(Time) +
Company(Time) + Battalion(Time) < 5 And Units(I, 1) = 0 And Units(I, 2)
= -1 And Time >= 1 Then
            J = 0
            Unitflag = True
            Do While J < 5
              If Hour(J + 5 * Time, 0) = Units(I, 0) Then Unitflag =
False
              If Hour(J + 5 * Time, 1) = 0 And (Units(I, 0) = 1 Or
Units(I, 0) = 2 Or Units(I, 0) = 3) Then Unitflag = False
              If Hour(J + 5 * Time, 1) = 4 And (Units(I, 0) = 5 Or
Units(I, 0) = 6 Or Units(I, 0) = 7) Then Unitflag = False
              If Hour(J + 5 * Time, 1) = 8 And (Units(I, 0) = 9 Or
Units(I, 0) = 10 Or Units(I, 0) = 11) Then Unitflag = False
              If Hour(J + 5 * Time, 1) = 12 And (Units(I, 0) = 13 Or
Units(I, 0) = 14 Or Units(I, 0) = 15) Then Unitflag = False
```

```
                K = J
                J = J + 1
              Loop
              J = 0
              Do While J < 5
                If Units(I, 2) = -1 And Unitflag = True And Hour(J + 5 *
Time, 0) = -2 And Hour(J + 5 * Time, 1) = -2 And Hour(J + 5 * Time, 2) =
-2 Then ' And Units(I, 4) + Saf_Row(Time) < Safs Then
                  Platoons(Time) = Platoons(Time) + 1
                  Platoons(Time - 1) = Platoons(Time - 1) + 1
                  'shut off unavailable blocks
                  Hour(J + 5 * Time, 0) = Units(I, 0)
                  Hour(J + 5 * (Time - 1), 0) = Units(I, 0)
                  Internal_Counter = 3
                  Do While Internal_Counter < 12
                    Hour(J + 5 * Time, Internal_Counter) = Units(I,
Internal_Counter)
                    Hour(J + 5 * (Time - 1), Internal_Counter) = Units(I,
Internal_Counter)
                    Internal_Counter = Internal_Counter + 1
                  Loop
                  Hour(J + 5 * Time, 1) = -1
                  Hour(J + 5 * Time, 2) = -1
                  Units(I, 2) = J + 5 * Time
                  Saf_Row(Time) = Saf_Row(Time) + Units(I, 4)
                  Saf_Row(Time - 1) = Saf_Row(Time - 1) + Units(I, 4)
                  J = 5
                End If 'Hour
                J = J + 1
                Loop
            End If ' the elseifs
          I = I - 1
          Loop   'i
If Schedule_Value >= 31 Or I = -1 Then Time = Time - 1
Loop 'time <hours
Resource_flag = True
MsgBox "formatting output"
  tbl_sched.Rows = 60

I = 0
Correction = 0
Do While I < Hours
  J = 0
  Saf_Total = 0
    Do While J < 5
          Sch_Str = ""
 ' Determine Search Criteria
        Echelon = "Platoon"
        context = "Platoon"
        If Hour(J + 5 * I, 3) >= 153 Then context = "Company"
        If Hour(J + 5 * I, 1) = 0 Or Hour(J + 5 * I, 1) = 4 Or Hour(J +
5 * I, 1) = 8 Or Hour(J + 5 * I, 1) = 12 Then
          Echelon = "Company"
          If Hour(J + 5 * I, 3) >= 153 Then
            context = "Battalion"
          Else
            context = "Company"
          End If
        End If
        If Hour(J + 5 * I, 2) = 16 Then
          Echelon = "Battalion"
```

```
    context = "Battalion"
  End If
  If Hour(J + 5 * I, 0) >= 0 Then
    Unit_Type = Unit_Str(Hour(J + 5 * I, 0))
  ElseIf Hour(J + 5 * I, 1) >= 0 Then
   Unit_Type = Unit_Str(Hour(J + 5 * I, 1))
 ElseIf Hour(J + 5 * I, 2) >= 0 Then
   Unit_Type = Unit_Str(Hour(J + 5 * I, 2))
 End If
  M = Hour(J + 5 * I, 3) Mod 9
Select Case M
  Case 0
    Event = "Movement to Contact" + Chr(13) + "Trained"
    Exercise = "Movement to Contact"
    Proficiency = "Trained"
  Case 1
    Event = "Movement to Contact" + Chr(13) + "Practice"
    Exercise = "Movement to Contact"
    Proficiency = "Practice"
  Case 2
    Event = "Movement to Contact" + Chr(13) + "Untrained"
    Exercise = "Movement to Contact"
    Proficiency = "Untrained"
  Case 3
    Event = "Attack" + Chr(13) + "Trained"
    Exercise = "Attack"
    Proficiency = "Trained"
  Case 4
    Event = "Attack" + Chr(13) + "Practice"
    Exercise = "Attack"
    Proficiency = "Practice"
  Case 5
    Event = "Attack" + Chr(13) + "Untrained"
    Exercise = "Attack"
    Proficiency = "Untrained"
Case 6
    Event = "Defend" + Chr(13) + "Trained"
    Exercise = "Defend"
    Proficiency = "Trained"
  Case 7
   Event = "Defend" + Chr(13) + "Practice"
   Exercise = "Defend"
    Proficiency = "Practice"
  Case 8
    Event = "Defend" + Chr(13) + "Untrained"
   Exercise = "Defend"
    Proficiency = "Untrained"
  Case Else
    Event = "Not Used"
    Exercise = ""
    Proficiency = ""
  End Select
    If Hour(J + 5 * I, 0) >= 0 Then
      Sch_Str = MAIN_Screen.UIC(Hour(J + 5 * I, 0)).Text
   ElseIf Hour(J + 5 * I, 1) >= 0 Then
      Sch_Str = MAIN_Screen.UIC(Hour(J + 5 * I, 1)).Text
    ElseIf Hour(J + 5 * I, 2) >= 0 Then
      Sch_Str = MAIN_Screen.UIC(Hour(J + 5 * I, 2)).Text
      Perf_Meas(I - Correction + 1) = 1#
    End If
    Unit_Str_Arr(J + 2, I - Correction + 1, 0) = Sch_Str
```

```
          Unit_Str_Arr(J + 2, I - Correction + 1, 1) = Exercise
          Unit_Str_Arr(J + 2, I - Correction + 1, 2) = Proficiency
          Unit_Str_Arr(J + 2, I - Correction + 1, 3) = Echelon
          Unit_Str_Arr(J + 2, I - Correction + 1, 4) = context
          Sch_Str1 = ""
          Saf_Total = Saf_Total + Hour(J + 5 * I, 4)
          If (J = 0 And Sch_Str <> "") Or J > 0 Then
            Sch_Str_Arr(J + 2, I - Correction + 1) = Sch_Str + Chr(13) +
Event
            Internal_Counter = 0
            Do While Internal_Counter < 8
              Saf_Array(J + 2, I - Correction + 1, Internal_Counter) =
Hour(J + 5 * I, Internal_Counter + 4)
              Internal_Counter = Internal_Counter + 1
            Loop
            tbl_sched.RowIndex = I - Correction + 1
            tbl_sched.ColumnIndex = J + 2
            Sch_Str = ""
          Else
          Correction = Correction + 1
          End If
      J = J + 1
    Loop 'j
    Sch_Str_Arr(1, I - Correction + 1) = Str(I - Correction + 1)



    Internal_Counter = 0
    Do While Internal_Counter < 7
      Saf_Array(7, I - Correction + 1, Internal_Counter) = 0
      Internal_counter2 = 2
        Do While Internal_counter2 < 7
          Saf_Array(7, I - Correction + 1, Internal_Counter) =
Saf_Array(7, I - Correction + 1, Internal_Counter) +
Saf_Array(Internal_counter2, I - Correction + 1, Internal_Counter)
          Internal_counter2 = Internal_counter2 + 1
        Loop
      Internal_Counter = Internal_Counter + 1
    Loop

    If Safs - Saf_Array(7, I - Correction + 1, 0) < 0 Then
      Sch_Str_Arr(7, I - Correction + 1) = "Not Enough Safs!!! " +
Chr(13) + "Safs over: " + Str(Saf_Array(7, I - Correction + 1, 0) -
Safs)
      Perf_Meas(I - Correction + 1) = 1#
      Resource_flag = False
    Else
      Sch_Str_Arr(7, I - Correction + 1) = "Safs: " + Str(Safs -
Saf_Array(7, I - Correction + 1, 0))
      Perf_Meas(I - Correction + 1) = CSng(Saf_Array(7, I - Correction +
1, 0) / Safs)
      If Unit_Str_Arr(2, I - Correction + 1, 3) = "Battalion" Then
Perf_Meas(I - Correction + 1) = 1#
    End If
    Sch_Str = ""
  I = I + 1
  Loop 'i
  tbl_sched.Rows = I - Correction
  Num_Rows_Tbl = I - Correction
  tbl_sched.ColumnIndex = 7
  I = 1
```

```
  Temp_PM = 0#
  Do While I < tbl_sched.Rows + 1
    Temp_PM = Perf_Meas(I) + Temp_PM
    I = I + 1
  Loop
  Temp_PM = Temp_PM / tbl_sched.Rows
  lbl_hours(0).Caption = "This Training Schedule Takes " +
Str(tbl_sched.Rows) + " Hours"
  lbl_hours(1).Caption = "Performance Measurement - " +
Str(Format(Temp_PM, "#.##"))
  tbl_sched.Refresh
' Set parameters so that the cell background color is red
' for untrained units
tbl_sched.ParamForeColor = INHERIT_COLOR
tbl_sched.ParamBackColor = GREEN
tbl_sched.ParamFontStyle = INHERIT_FONT
tbl_sched.ParamStatus = -1
' Add a pattern to the grid pattern matching table
'tbl_sched.AddRegexAttr = Chr(8)
tbl_sched.AddRegexAttr = Chr(13) + "Tra"
tbl_sched.ParamForeColor = INHERIT_COLOR
tbl_sched.ParamBackColor = YELLOW
tbl_sched.ParamFontStyle = INHERIT_FONT
tbl_sched.ParamStatus = -1
' Add a pattern to the grid pattern matching table
tbl_sched.AddRegexAttr = "Practice"
tbl_sched.ParamForeColor = INHERIT_COLOR
tbl_sched.ParamBackColor = RED
tbl_sched.ParamFontStyle = INHERIT_FONT
tbl_sched.ParamStatus = -1
' Add a pattern to the grid pattern matching table
tbl_sched.AddRegexAttr = Chr(13) + "Unt"
tbl_sched.ParamForeColor = INHERIT_COLOR
tbl_sched.ParamBackColor = LT_GRAY
tbl_sched.ParamFontStyle = INHERIT_FONT
tbl_sched.ParamStatus = -1
tbl_sched.AddRegexAttr = "Not Used"
tbl_sched.SetStatusAttr = 1
'  Keep this next command last
tbl_sched.Refresh
If Resource_flag = flase Then
  cmd_Adjust.visible = True
  res_label.visible = True
  line1.visible = True
End If

frm_Sch_List.Show
Else
  MsgBox "Nothing has been placed on the Event List - Please Return and
enter some events to schedule."
End If
End Sub

Private Sub mnu_Print_Click()
Adjust_Res
End Sub

Private Sub mnu_Redo_Click()
Unload Me
End Sub
```

```
Public Sub tbl_sched_DblClick()
Dim Row, column As Integer

column = tbl_sched.ColumnAtPoint
Row = tbl_sched.RowAtPoint
If Saf_Array(column, Row, 0) > 0 And column < 7 Then
  msg = "Unit: " + Unit_Str_Arr(column, Row, 0) + Chr(13) + "Exercise: "
+ Unit_Str_Arr(column, Row, 1) + Chr(13) + "Training Status: " +
Unit_Str_Arr(column, Row, 2) + Chr(13) + "This will be conducted as a "
+ Unit_Str_Arr(column, Row, 3) + " within a " + Unit_Str_Arr(column,
Row, 4) + Chr(13) + "SAFs used for this event are: " +
Str(Saf_Array(column, Row, 0)) + Chr(13) + "Lower Bound for SAFs: " +
Str(Saf_Array(column, Row, 7)) + Chr(13) + "M1s used are: " +
Str(Saf_Array(column, Row, 1)) + Chr(13) + "M2s used are: " +
Str(Saf_Array(column, Row, 2)) + Chr(13) + "DIMs used are: " +
Str(Saf_Array(column, Row, 3)) + Chr(13) + "APCs used are:" +
Str(Saf_Array(column, Row, 4)) + Chr(13) + "HVs used are: " +
Str(Saf_Array(column, Row, 5))
ElseIf Saf_Array(column, Row, 0) > 0 And column = 7 Then
  msg = "SAFs used for this hour are: " + Str(Saf_Array(column, Row, 0))
+ Chr(13) + "M1s used are: " + Str(Saf_Array(column, Row, 1)) + Chr(13)
+ "M2s used are: " + Str(Saf_Array(column, Row, 2)) + Chr(13) + "DIMs
used are: " + Str(Saf_Array(column, Row, 3)) + Chr(13) + "APCs used
are:" + Str(Saf_Array(column, Row, 4)) + Chr(13) + "HVs used are: " +
Str(Saf_Array(column, Row, 5)) + Chr(13) + "Efficiency for this hour is:
" + Str(Format(Perf_Meas(Row), "#.##")) 'Saf_Array(Column, Row, 0) /
Safs)
Else
msg = "Sorry, No event is scheduled here"
  End If


Style = vbOKOnly ' Define buttons.
Title = "Summary"

response = MsgBox(msg, Style, Title)

End Sub

Public Sub tbl_sched_Fetch(Row As Long, Col As Integer, Value As String)

  Value = Sch_Str_Arr(Col, Row)

End Sub

Public Function SAF_FCN(Pass_Event, Pass_Unit)
dta_scen.Refresh
 ' Determine Search Criteria
        Echelon = "Platoon"
        context = "Platoon"

        If Pass_Event >= 153 Then context = "Company"
        If Pass_Unit = 0 Or Pass_Unit = 4 Or Pass_Unit = 8 Or Pass_Unit
= 12 Then
          Echelon = "Company"
          If Pass_Event >= 153 Then
            context = "Battalion"
          Else
            context = "Company"
          End If
        End If
```

```
        If Pass_Unit = 16 Then
          Echelon = "Battalion"
          context = "Battalion"
        End If


         Unit_Type = Unit_Str(Pass_Unit)

        M = Pass_Event Mod 9


      Select Case M
        Case 0
          Event = "Movement to Contact - Trained"
          Exercise = "Movement to Contact"
          Proficiency = "Trained"
        Case 1
          Event = "Movement to Contact - Practice"
          Exercise = "Movement to Contact"
          Proficiency = "Practice"
        Case 2
          Event = "Movement to Contact - Untrained"
          Exercise = "Movement to Contact"
          Proficiency = "Untrained"
        Case 3
          Event = "Attack - Trained"
          Exercise = "Attack"
          Proficiency = "Trained"
        Case 4
          Event = "Attack - Practice"
          Exercise = "Attack"
          Proficiency = "Practice"
        Case 5
          Event = "Attack - Untrained"
          Exercise = "Attack"
          Proficiency = "Untrained"
        Case 6
          Event = "Defend - Trained"
          Exercise = "Defend"
          Proficiency = "Trained"
        Case 7
          Event = "Defend - Practice"
          Exercise = "Defend"
          Proficiency = "Practice"
        Case 8
          Event = "Defend - Untrained"
          Exercise = "Defend"
          Proficiency = "Untrained"
        Case Else
          Event = "Not Used"
          Exercise = ""
          Proficiency = ""
        End Select
      dta_scen.Recordset.MoveFirst
        Do While dta_scen.Recordset.EOF <> True
          If Exercise = dta_scen.Recordset.Fields(0) And Proficiency =
dta_scen.Recordset.Fields(1) And Echelon = dta_scen.Recordset.Fields(2)
And context = dta_scen.Recordset.Fields(3) And Unit_Type =
dta_scen.Recordset.Fields(5) Then
            SAF_FCN = dta_scen.Recordset.Fields(6) +
dta_scen.Recordset.Fields(7)
```

```
            dta_scen.Recordset.MoveLast
          Else
            dta_scen.Recordset.MoveNext
          End If
        Loop

End Function

Public Function Unit_FCN(Pass_Event, Pass_Unit, Pass_Res)
Dim I, J As Integer
Dim Plt(4) As Integer
Dim Unit_Type_Plt(3) As String
Dim Unit_Type, Echelon, Unit_Involve As String
Dim Unit_Involve_Plt(3) As String
Dim Co As Boolean
Co = True
Echelon = "Platoon"
If Pass_Unit = 16 Then Echelon = "Battalion"
  Unit_Type = Unit_Str(Pass_Unit)
  Unit_Involve = "All"
  If Pass_Unit = 0 Or Pass_Unit = 4 Or Pass_Unit = 8 Or Pass_Unit = 12
Then ' got  a company
    Echelon = "Company"
    Co = False
    Unit_Involve = "All"
'     Echelon = "Platoon"
    J = 0
    Do While J < 3
      Plt(J) = Pass_Unit + J - Int(Pass_Unit / 4)
      Unit_Type_Plt(J) = Unit_Str(Pass_Unit + J + 1)
       If MAIN_Screen.Plt_Entire(Plt(J)).Value = True Then
          Unit_Involve_Plt(J) = "All"
       ElseIf MAIN_Screen.Plt_Minus(Plt(J)).Value = True Then
          Unit_Involve_Plt(J) = "PL"
       Else
          Unit_Involve_Plt(J) = "None"
       End If
   J = J + 1
   Loop
End If
       dta_Blue.Recordset.MoveFirst
        Do While dta_Blue.Recordset.EOF <> True
          If Echelon = dta_Blue.Recordset.Fields(1) And Unit_Type =
dta_Blue.Recordset.Fields(0) And Unit_Involve =
dta_Blue.Recordset.Fields(2) Then
            I = dta_Blue.Recordset.Fields(Pass_Res - 2)
            dta_Blue.Recordset.MoveLast
          Else
            dta_Blue.Recordset.MoveNext
          End If
        Loop

J = 0
If Company = False Then
' Do the platoons
Do While J < 3
       dta_Blue.Recordset.MoveFirst
        Do While dta_Blue.Recordset.EOF <> True
          If dta_Blue.Recordset.Fields(1) = "Platoon" And
Unit_Involve_Plt(J) = dta_Blue.Recordset.Fields(2) And Unit_Type_Plt(J)
= dta_Blue.Recordset.Fields(0) Then
```

```
            I = I + dta_Blue.Recordset.Fields(Pass_Res - 2)
            dta_Blue.Recordset.MoveLast
          Else
            dta_Blue.Recordset.MoveNext
          End If
        Loop
 J = J + 1
 Loop
 End If

 Unit_FCN = I

End Function

Private Sub tbl_sched_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)
tbl_sched.PointX = X
tbl_sched.PointY = Y
End Sub

Public Function SAF_LB(Pass_Event, Pass_Unit)
dta_scen.Refresh
 ' Determine Search Criteria
        Echelon = "Platoon"
        context = "Platoon"

        If Pass_Event >= 153 Then context = "Company"
        If Pass_Unit = 0 Or Pass_Unit = 4 Or Pass_Unit = 8 Or Pass_Unit
= 12 Then
          Echelon = "Company"
          If Pass_Event >= 153 Then
            context = "Battalion"
          Else
            context = "Company"
          End If
        End If
        If Pass_Unit = 16 Then
          Echelon = "Battalion"
          context = "Battalion"
        End If


         Unit_Type = Unit_Str(Pass_Unit)


        M = Pass_Event Mod 9

Proficiency = "Untrained"
      Select Case M
        Case 0
          Event = "Movement to Contact - Trained"
          Exercise = "Movement to Contact"

        Case 1
          Event = "Movement to Contact - Practice"
          Exercise = "Movement to Contact"

        Case 2
          Event = "Movement to Contact - Untrained"
          Exercise = "Movement to Contact"
```

```
        Case 3
          Event = "Attack - Trained"
          Exercise = "Attack"

        Case 4
          Event = "Attack - Practice"
          Exercise = "Attack"

        Case 5
          Event = "Attack - Untrained"
          Exercise = "Attack"

        Case 6
          Event = "Defend - Trained"
          Exercise = "Defend"

        Case 7
          Event = "Defend - Practice"
          Exercise = "Defend"

        Case 8
          Event = "Defend - Untrained"
          Exercise = "Defend"

        Case Else
          Event = "Not Used"
          Exercise = ""
          Proficiency = ""
        End Select
      dta_scen.Recordset.MoveFirst
        Do While dta_scen.Recordset.EOF <> True
          If Exercise = dta_scen.Recordset.Fields(0) And Proficiency =
dta_scen.Recordset.Fields(1) And Echelon = dta_scen.Recordset.Fields(2)
And context = dta_scen.Recordset.Fields(3) And Unit_Type =
dta_scen.Recordset.Fields(5) Then
            SAF_LB = dta_scen.Recordset.Fields(6) +
dta_scen.Recordset.Fields(7)
            dta_scen.Recordset.MoveLast
          Else
            dta_scen.Recordset.MoveNext
          End If
        Loop
End Function

Public Sub Adjust_Res()
Dim I, J, K, L, Row, column, Saf_Total As Integer
Dim temp_ppm As Single
Row = Num_Rows_Tbl
  Do While Row > 0
  Do Until Saf_Array(7, Row, 0) <= Safs
      column = 2
      Do While column < 7
        If Saf_Array(column, Row, 0) > Saf_Array(column, Row, 7) Then
          Saf_Array(column, Row, 0) = Saf_Array(column, Row, 0) - 1
          If Row < Num_Rows_Tbl Then
            If Unit_Str_Arr(column, Row + 1, 0) = Unit_Str_Arr(column,
Row, 0) And Unit_Str_Arr(column, Row + 1, 1) = Unit_Str_Arr(column, Row,
1) Then Saf_Array(column, Row + 1, 0) = Saf_Array(column, Row + 1, 0) -
1
          End If
          If Row < Num_Rows_Tbl - 1 Then
```

```
            If Unit_Str_Arr(column, Row + 2, 0) = Unit_Str_Arr(column,
Row, 0) And Unit_Str_Arr(column, Row + 2, 1) = Unit_Str_Arr(column, Row,
1) Then Saf_Array(column, Row + 2, 0) = Saf_Array(column, Row + 2, 0) -
1
          End If
          If Row > 1 Then
            If Unit_Str_Arr(column, Row - 1, 0) = Unit_Str_Arr(column,
Row, 0) And Unit_Str_Arr(column, Row - 1, 1) = Unit_Str_Arr(column, Row,
1) Then Saf_Array(column, Row - 1, 0) = Saf_Array(column, Row - 1, 0) -
1
          End If
          If Row > 2 Then
            If Unit_Str_Arr(column, Row - 2, 0) = Unit_Str_Arr(column,
Row, 0) And Unit_Str_Arr(column, Row - 2, 1) = Unit_Str_Arr(column, Row,
1) Then Saf_Array(column, Row - 2, 0) = Saf_Array(column, Row - 2, 0) -
1
          End If
        End If
        column = column + 1
      Loop
       column = 2
      Saf_Array(7, Row, 0) = 0
        Do While column < 7
          Saf_Array(7, Row, 0) = Saf_Array(7, Row, 0) +
Saf_Array(column, Row, 0)
          column = column + 1
        Loop
    Loop
    Row = Row - 1
  Loop
Row = 1
  Do While Row < Num_Rows_Tbl + 1
      column = 2
      Saf_Array(7, Row, 0) = 0
        Do While column < 7
          Saf_Array(7, Row, 0) = Saf_Array(7, Row, 0) +
Saf_Array(column, Row, 0)
          column = column + 1
        Loop
      Sch_Str_Arr(7, Row) = "Safs: " + Str(Safs - Saf_Array(7, Row, 0))
      Perf_Meas(Row) = Format(CSng(Saf_Array(7, Row, 0) / Safs), "#.##")
      If Unit_Str_Arr(2, Row, 3) = "Battalion" Then Perf_Meas(Row) = 1
      Row = Row + 1
   Loop
  I = 1
  temp_ppm = 0#
  Do While I < tbl_sched.Rows + 1
    temp_ppm = Perf_Meas(I) + temp_ppm
    I = I + 1
  Loop
  temp_ppm = CSng(temp_ppm / CSng(tbl_sched.Rows))
 frm_Sch_List.lbl_hours(1).Caption = "Performance Measurement - " +
Str(Format(temp_ppm, "#.##"))


tbl_sched.Refresh

End Sub

Attribute VB_Name = "frm_Sel_event"
```

```
Attribute VB_Creatable = False
Attribute VB_Exposed = False

Private Sub Check1_Click(Index As Integer)
Dim I As Integer

If Index Mod 2 = 0 And Check1(Index).Value = 1 Then Check1(Index +
1).Value = 0
'If Index Mod 2 = 0 And Check1(Index).Value = 0 Then Check1(Index +
1).Value = 1
If Index Mod 2 = 1 And Check1(Index).Value = 1 Then Check1(Index -
1).Value = 0
'If Index Mod 2 = 1 And Check1(Index).Value = 0 Then Check1(Index -
1).Value = 1

I = 0
Do While I < 306
If Schedule(I, 1) = Index Then
Schedule(I, 0) = Check1(Index).Value
End If
I = I + 1
Loop
End Sub

Private Sub Command1_Click()
Dim I As Integer
I = 0
Do While I < 306
Schedule(I, 1) = -1
I = I + 1
Loop
Unload Me
End Sub

Public Sub Untrained()
Dim I, J, K, L, M, Correction As Integer
Dim Str As String
Correction = 0
K = 0
Do While K < 25
Label1(K).visible = True
Label1(K + 25).visible = True
Check1(2 * K).visible = True
Check1(2 * K + 1).visible = True
Check1(2 * K).Enabled = True
Check1(2 * K + 1).Enabled = True
Check1(2 * K).Value = 0
Check1(2 * K + 1).Value = 0
Check1(2 * K).Caption = "Platoon"
Check1(2 * K + 1).Caption = "Company"
K = K + 1
Loop

K = 0 ' Block Counter'
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
```

```
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 11
J = 1 ' Unit Counter
L = 0

Do While I < 144

If MAIN_Screen.txt_Priority(J) Like Str And
MAIN_Screen.unit_task(I).Value = True Then
  Label1(K).Caption = "Untrained; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I, 0)
    Schedule(I, 1) = 2 * K
    Schedule(I + 153, 1) = 2 * K + 1
    Schedule(I, 2) = Schedule(I, 1) + Correction
    Schedule(I + 153, 2) = Schedule(I + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I, 2) = 999
    Schedule(I + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 11
J = 1
L = 0

Do While I < 144

If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
1).Value = True Then
  Label1(K).Caption = "Practice; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
```

```
    If MAIN_Screen.Plt_Entire(L) = True Then
      Check1(2 * K).Value = Schedule(I - 1, 0)
      Schedule(I - 1, 1) = 2 * K
      Schedule(I - 1 + 153, 1) = 2 * K + 1
      Schedule(I - 1, 2) = Schedule(I - 1, 1) + Correction
      Schedule(I - 1 + 153, 2) = Schedule(I - 1 + 153, 1) + Correction
      Check1(2 * K + 1).Value = Schedule(I - 1 + 153, 0)
      Check1(2 * K).Enabled = True
      Check1(2 * K + 1).Enabled = True
    Else
      Schedule(I - 1, 2) = 999
      Schedule(I - 1 + 153, 2) = 999
      Check1(2 * K).Enabled = False
      Check1(2 * K + 1).Enabled = False
    End If
    K = K + 1
  End If

  J = J + 1
  L = L + 1
  I = I + 9
  If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
    I = I + 9
    J = J + 1
  End If
  Loop

  M = M + 1
  Loop 'm
  M = 0 ' Priority Counter

  Do While M < 5

  If M = 0 Then Str = "1"
  If M = 1 Then Str = "2"
  If M = 2 Then Str = "3"
  If M = 3 Then Str = "4"
  If M = 4 Then Str = "5"

  I = 11
  J = 1
  L = 0

  Do While I < 144

  If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
  2).Value = True Then
    Label1(K).Caption = "Trained; Priority = " + Str
    Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
    If MAIN_Screen.Plt_Entire(L) = True Then
      Check1(2 * K).Value = Schedule(I - 2, 0)
      Schedule(I - 2, 1) = 2 * K
      Schedule(I - 2 + 153, 1) = 2 * K + 1
      Schedule(I - 2, 2) = Schedule(I - 2, 1) + Correction
      Schedule(I - 2 + 153, 2) = Schedule(I - 2 + 153, 1) + Correction
      Check1(2 * K + 1).Value = Schedule(I - 2 + 153, 0)
      Check1(2 * K).Enabled = True
      Check1(2 * K + 1).Enabled = True
    Else
      Schedule(I - 2, 2) = 999
      Schedule(I - 2 + 153, 2) = 999
```

```
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If

J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
Counter(0) = 2 * K
Do While K < 25
Label1(K).visible = False
Label1(K + 25).visible = False
Check1(2 * K).visible = False
Check1(2 * K + 1).visible = False
K = K + 1
Loop

End Sub

Public Sub Practice()
Dim I, J, K, L, M, Correction As Integer
Dim Str As String
Correction = 50
K = 0
Do While K < 25
Label1(K).visible = True
Label1(K + 25).visible = True
Check1(2 * K).visible = True
Check1(2 * K + 1).visible = True
Check1(2 * K).Enabled = True
Check1(2 * K + 1).Enabled = True
Check1(2 * K).Value = 0
Check1(2 * K + 1).Value = 0
Check1(2 * K).Caption = "Platoon"
Check1(2 * K + 1).Caption = "Company"
K = K + 1
Loop

K = 0 ' Block Counter'
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"


I = 14
J = 1 ' Unit Counter
```

```
L = 0

Do While I < 144

If MAIN_Screen.txt_Priority(J) Like Str And
MAIN_Screen.unit_task(I).Value = True Then
  Label1(K).Caption = "Untrained; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I, 0)
    Schedule(I, 1) = 2 * K
    Schedule(I + 153, 1) = 2 * K + 1
    Schedule(I, 2) = Schedule(I, 1) + Correction
    Schedule(I + 153, 2) = Schedule(I + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I, 2) = 999
    Schedule(I + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 14
J = 1
L = 0

Do While I < 144

If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
1).Value = True Then
  Label1(K).Caption = "Practice; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I - 1, 0)
    Schedule(I - 1, 1) = 2 * K
    Schedule(I - 1 + 153, 1) = 2 * K + 1
    Schedule(I - 1, 2) = Schedule(I - 1, 1) + Correction
```

```
    Schedule(I - 1 + 153, 2) = Schedule(I - 1 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 1 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 1, 2) = 999
    Schedule(I - 1 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If

J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 14
J = 1
L = 0

Do While I < 144

If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
2).Value = True Then
  Label1(K).Caption = "Trained; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I - 2, 0)
    Schedule(I - 2, 1) = 2 * K
    Schedule(I - 2 + 153, 1) = 2 * K + 1
    Schedule(I - 2, 2) = Schedule(I - 2, 1) + Correction
    Schedule(I - 2 + 153, 2) = Schedule(I - 2 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 2 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 2, 2) = 999
    Schedule(I - 2 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
```

```
J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
Counter(0) = 2 * K
Do While K < 25
Label1(K).visible = False
Label1(K + 25).visible = False
Check1(2 * K).visible = False
Check1(2 * K + 1).visible = False
K = K + 1
Loop

End Sub

Public Sub Trained()
Dim I, J, K, L, M, Correction As Integer
Dim Str As String
Correction = 100
K = 0
Do While K < 25
Label1(K).visible = True
Label1(K + 25).visible = True
Check1(2 * K).visible = True
Check1(2 * K + 1).visible = True
Check1(2 * K).Enabled = True
Check1(2 * K + 1).Enabled = True
Check1(2 * K).Value = 0
Check1(2 * K + 1).Value = 0
Check1(2 * K).Caption = "Platoon"
Check1(2 * K + 1).Caption = "Company"
K = K + 1
Loop

K = 0 ' Block Counter'
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"


I = 17
J = 1 ' Unit Counter
L = 0

Do While I < 144
```

```
If MAIN_Screen.txt_Priority(J) Like Str And
MAIN_Screen.unit_task(I).Value = True Then
  Label1(K).Caption = "Untrained; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I, 0)
    Schedule(I, 1) = 2 * K
    Schedule(I + 153, 1) = 2 * K + 1
    Schedule(I, 2) = Schedule(I, 1) + Correction
    Schedule(I + 153, 2) = Schedule(I + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I, 2) = 999
    Schedule(I + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 17
J = 1
L = 0

Do While I < 144


If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
1).Value = True Then
  Label1(K).Caption = "Practice; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I - 1, 0)
    Schedule(I - 1, 1) = 2 * K
    Schedule(I - 1 + 153, 1) = 2 * K + 1
    Schedule(I - 1, 2) = Schedule(I - 1, 1) + Correction
    Schedule(I - 1 + 153, 2) = Schedule(I - 1 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 1 + 153, 0)
```

```
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 1, 2) = 999
    Schedule(I - 1 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If

J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
M = 0 ' Priority Counter

Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"



I = 17
J = 1
L = 0

Do While I < 144

If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
2).Value = True Then
  Label1(K).Caption = "Trained; Priority = " + Str
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.Plt_Entire(L) = True Then
    Check1(2 * K).Value = Schedule(I - 2, 0)
    Schedule(I - 2, 1) = 2 * K
    Schedule(I - 2 + 153, 1) = 2 * K + 1
    Schedule(I - 2, 2) = Schedule(I - 2, 1) + Correction
    Schedule(I - 2 + 153, 2) = Schedule(I - 2 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 2 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 2, 2) = 999
    Schedule(I - 2 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
```

```
End If

J = J + 1
L = L + 1
I = I + 9
If J = 4 Or J = 8 Or J = 12 Or J = 16 Then
  I = I + 9
  J = J + 1
End If
Loop

M = M + 1
Loop 'm
Counter(0) = 2 * K
Do While K < 25
Label1(K).visible = False
Label1(K + 25).visible = False
Check1(2 * K).visible = False
Check1(2 * K + 1).visible = False
K = K + 1
Loop

End Sub

Public Sub Company()
Dim I, J, K, L, M, Correction As Integer
Dim Str As String
Correction = 150
K = 0
Do While K < 25
Label1(K).visible = True
Label1(K + 25).visible = True
Check1(2 * K).visible = True
Check1(2 * K + 1).visible = True
Check1(2 * K).Value = 0
Check1(2 * K + 1).Value = 0
Check1(2 * K).Caption = "Company"
Check1(2 * K + 1).Caption = "Battalion"
K = K + 1
Loop

K = 0 ' Block Counter'
M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"
I = 2
J = 0 ' Unit Counter
L = 0

Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And
MAIN_Screen.unit_task(I).Value = True Then
  Label1(K).Caption = "U - Movement to Contact"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
```

```
      Check1(2 * K).Value = Schedule(I, 0)
      Schedule(I, 1) = 2 * K
      Schedule(I + 153, 1) = 2 * K + 1
      Schedule(I, 2) = Schedule(I, 1) + Correction
      Schedule(I + 153, 2) = Schedule(I + 153, 1) + Correction
      Check1(2 * K + 1).Value = Schedule(I + 153, 0)
      Check1(2 * K).Enabled = True
      Check1(2 * K + 1).Enabled = True
    Else
      Schedule(I, 2) = 999
      Schedule(I + 153, 2) = 999
      Check1(2 * K).Enabled = False
      Check1(2 * K + 1).Enabled = False
    End If
  K = K + 1
  End If
  L = L + 1
  J = J + 4
  I = I + 36
  Loop

  M = M + 1
  Loop 'M
  M = 0 ' Priority Counter
  Do While M < 5

  If M = 0 Then Str = "1"
  If M = 1 Then Str = "2"
  If M = 2 Then Str = "3"
  If M = 3 Then Str = "4"
  If M = 4 Then Str = "5"

  I = 2
  J = 0 ' Unit Counter
  L = 0
  Do While I < 144
  If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
  1).Value = True Then
    Label1(K).Caption = "P - Movement to Contact"
    Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
    If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
  Then
      Check1(2 * K).Value = Schedule(I - 1, 0)
      Schedule(I - 1, 1) = 2 * K
      Schedule(I - 1 + 153, 1) = 2 * K + 1
      Schedule(I - 1, 2) = Schedule(I - 1, 1) + Correction
      Schedule(I - 1 + 153, 2) = Schedule(I - 1 + 153, 1) + Correction
      Check1(2 * K + 1).Value = Schedule(I - 1 + 153, 0)
      Check1(2 * K).Enabled = True
      Check1(2 * K + 1).Enabled = True
    Else
      Schedule(I - 1, 2) = 999
      Schedule(I - 1 + 153, 2) = 999
      Check1(2 * K).Enabled = False
      Check1(2 * K + 1).Enabled = False
    End If
  K = K + 1
  End If
  L = L + 1
  J = J + 4
  I = I + 36
```

```
Loop

M = M + 1
Loop 'M
M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 2
J = 0 ' Unit Counter
L = 0
Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
2).Value = True Then
  Label1(K).Caption = "T - Movement to Contact"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
    Check1(2 * K).Value = Schedule(I - 2, 0)
    Schedule(I - 2, 1) = 2 * K
    Schedule(I - 2 + 153, 1) = 2 * K + 1
    Schedule(I - 2, 2) = Schedule(I - 2, 1) + Correction
    Schedule(I - 2 + 153, 2) = Schedule(I - 2 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 2 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 2, 2) = 999
    Schedule(I - 2 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
K = K + 1
End If
L = L + 1
J = J + 4
I = I + 36

Loop
M = M + 1
Loop 'M

M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"
I = 5
J = 0 ' Unit Counter
L = 0

Do While I < 144
```

```
If MAIN_Screen.txt_Priority(J) Like Str And
MAIN_Screen.unit_task(I).Value = True Then
  Label1(K).Caption = "U - Attack"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
    Check1(2 * K).Value = Schedule(I, 0)
    Schedule(I, 1) = 2 * K
    Schedule(I + 153, 1) = 2 * K + 1
    Schedule(I, 2) = Schedule(I, 1) + Correction
    Schedule(I + 153, 2) = Schedule(I + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I, 2) = 999
    Schedule(I + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1

End If

L = L + 1
J = J + 4
I = I + 36
Loop

M = M + 1
Loop 'M
M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 5
J = 0 ' Unit Counter
L = 0

Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
1).Value = True Then
  Label1(K).Caption = "P - Attack"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
    Check1(2 * K).Value = Schedule(I - 1, 0)
    Schedule(I - 1, 1) = 2 * K
    Schedule(I - 1 + 153, 1) = 2 * K + 1
    Schedule(I - 1, 2) = Schedule(I - 1, 1) + Correction
    Schedule(I - 1 + 153, 2) = Schedule(I - 1 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 1 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 1, 2) = 999
```

```
    Schedule(I - 1 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If

L = L + 1
J = J + 4
I = I + 36
Loop

M = M + 1
Loop 'M
M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 5
J = 0 ' Unit Counter
L = 0
Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
2).Value = True Then
  Label1(K).Caption = "T - Attack"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
    Check1(2 * K).Value = Schedule(I - 2, 0)
    Schedule(I - 2, 1) = 2 * K
    Schedule(I - 2 + 153, 1) = 2 * K + 1
    Schedule(I - 2, 2) = Schedule(I - 2, 1) + Correction
    Schedule(I - 2 + 153, 2) = Schedule(I - 2 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 2 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 2, 2) = 999
    Schedule(I - 2 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
L = L + 1
J = J + 4
I = I + 36

Loop
M = M + 1
Loop 'M

M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
```

```
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"
I = 8
J = 0 ' Unit Counter
L = 0

Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And
MAIN_Screen.unit_task(I).Value = True Then
  Label1(K).Caption = "U - Defend "
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
  If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
    Check1(2 * K).Value = Schedule(I, 0)
    Schedule(I, 1) = 2 * K
    Schedule(I + 153, 1) = 2 * K + 1
    Schedule(I, 2) = Schedule(I, 1) + Correction
    Schedule(I + 153, 2) = Schedule(I + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I, 2) = 999
    Schedule(I + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If

  K = K + 1

End If
L = L + 1
J = J + 4
I = I + 36


Loop

M = M + 1
Loop 'M
M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 8
J = 0 ' Unit Counter
L = 0
Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
1).Value = True Then
  Label1(K).Caption = "P - Defend"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
   If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
```

```
    Check1(2 * K).Value = Schedule(I - 1, 0)
    Schedule(I - 1, 1) = 2 * K
    Schedule(I - 1 + 153, 1) = 2 * K + 1
    Schedule(I - 1, 2) = Schedule(I - 1, 1) + Correction
    Schedule(I - 1 + 153, 2) = Schedule(I - 1 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 1 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 1, 2) = 999
    Schedule(I - 1 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
L = L + 1
J = J + 4
I = I + 36
Loop

M = M + 1
Loop 'M
M = 0 ' Priority Counter
Do While M < 5

If M = 0 Then Str = "1"
If M = 1 Then Str = "2"
If M = 2 Then Str = "3"
If M = 3 Then Str = "4"
If M = 4 Then Str = "5"

I = 8
J = 0 ' Unit Counter
L = 0
Do While I < 144
If MAIN_Screen.txt_Priority(J) Like Str And MAIN_Screen.unit_task(I -
2).Value = True Then
  Label1(K).Caption = "T - Defend"
  Label1(K + 25).Caption = MAIN_Screen.UIC(J).Text
    If MAIN_Screen.CO_Entire(L) = True Or MAIN_Screen.CO_Staff(L) = True
Then
    Check1(2 * K).Value = Schedule(I - 2, 0)
    Schedule(I - 2, 1) = 2 * K
    Schedule(I - 2 + 153, 1) = 2 * K + 1
    Schedule(I - 2, 2) = Schedule(I - 2, 1) + Correction
    Schedule(I - 2 + 153, 2) = Schedule(I - 2 + 153, 1) + Correction
    Check1(2 * K + 1).Value = Schedule(I - 2 + 153, 0)
    Check1(2 * K).Enabled = True
    Check1(2 * K + 1).Enabled = True
  Else
    Schedule(I - 2, 2) = 999
    Schedule(I - 2 + 153, 2) = 999
    Check1(2 * K).Enabled = False
    Check1(2 * K + 1).Enabled = False
  End If
  K = K + 1
End If
L = L + 1
J = J + 4
I = I + 36
```

```
Loop
M = M + 1
Loop 'M
Counter(3) = 2 * K + 1
Do While K < 25
Label1(K).visible = False
Label1(K + 25).visible = False
Check1(2 * K).visible = False
Check1(2 * K + 1).visible = False
K = K + 1
Loop
End Sub

Public Sub Battalion()
Dim Correction, K As Integer
K = 0
Correction = 200
Do While K < 25
Label1(K).visible = True
Label1(K + 25).visible = True
Check1(2 * K).visible = True
Check1(2 * K + 1).visible = False
Check1(2 * K).Value = 0
Check1(2 * K + 1).Value = 0
Check1(2 * K).Caption = "Battalion"
Check1(2 * K + 1).Enabled = False
K = K + 1
Loop
K = 0 ' Block Counter'
If MAIN_Screen.unit_task(146).Value = True Then
  Label1(K).Caption = "U - Movement to Contact "
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
  If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
    Check1(2 * K).Value = Schedule(146, 0)
    Schedule(146, 1) = 2 * K
    Schedule(146, 2) = Schedule(146, 1) + Correction
    Check1(2 * K).Enabled = True
  Else
    Check1(2 * K).Enabled = False
    Schedule(146, 2) = 999
  End If
K = K + 1
End If
If MAIN_Screen.unit_task(145).Value = True Then
  Label1(K).Caption = "P - Movement to Contact"
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
    If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True
Then
    Check1(2 * K).Value = Schedule(145, 0)
    Schedule(145, 1) = 2 * K
    Schedule(145, 2) = Schedule(145, 1) + Correction
    Check1(2 * K).Enabled = True
  Else
  Check1(2 * K).Enabled = False
  Schedule(145, 2) = 999
  End If
  K = K + 1
End If
If MAIN_Screen.unit_task(144).Value = True Then
  Label1(K).Caption = "T - Movement to Contact"
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
```

```
  If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
    Check1(2 * K).Value = Schedule(144, 0)
    Schedule(144, 1) = 2 * K
    Schedule(144, 2) = Schedule(144, 1) + Correction
    Check1(2 * K).Enabled = True
  Else
  Schedule(144, 2) = 999
  Check1(2 * K).Enabled = False
  End If
  K = K + 1
End If
If MAIN_Screen.unit_task(149).Value = True Then
  Label1(K).Caption = "U - Attack"
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
  If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
    Check1(2 * K).Value = Schedule(149, 0)
    Schedule(149, 1) = 2 * K
    Schedule(149, 2) = Schedule(149, 1) + Correction
    Check1(2 * K).Enabled = True
  Else
  Check1(2 * K).Enabled = False
  Schedule(149, 2) = 999
  End If
K = K + 1
End If
If MAIN_Screen.unit_task(148).Value = True Then
  Label1(K).Caption = "P - Attack"
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
  If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
    Check1(2 * K).Value = Schedule(148, 0)
    Schedule(148, 1) = 2 * K
    Schedule(148, 2) = Schedule(148, 1) + Correction
    Check1(2 * K).Enabled = True
  Else
  Schedule(148, 2) = 999
  Check1(2 * K).Enabled = False
  End If
  K = K + 1
End If
If MAIN_Screen.unit_task(147).Value = True Then
  Label1(K).Caption = "T - Attack"
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
   If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
    Check1(2 * K).Value = Schedule(147, 0)
    Schedule(147, 1) = 2 * K
    Schedule(147, 2) = Schedule(147, 1) + Correction
    Check1(2 * K).Enabled = True
  Else
  Schedule(147, 2) = 999
  Check1(2 * K).Enabled = False
  End If
  K = K + 1
End If
If MAIN_Screen.unit_task(152).Value = True Then
  Label1(K).Caption = "U - Defend"
  Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
  If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
  Check1(2 * K).Value = Schedule(152, 0)
    Schedule(152, 1) = 2 * K
    Schedule(152, 2) = Schedule(152, 1) + Correction
    Check1(2 * K).Enabled = True
```

```
    Else
    Check1(2 * K).Enabled = False
    Schedule(152, 2) = 999
    End If
  K = K + 1
  End If
  If MAIN_Screen.unit_task(151).Value = True Then
    Label1(K).Caption = "P - Defend"
    Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
    If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
      Check1(2 * K).Value = Schedule(151, 0)
      Schedule(151, 1) = 2 * K
      Schedule(151, 2) = Schedule(151, 1) + Correction
      Check1(2 * K).Enabled = True
    Else
    Schedule(151, 2) = 999
    Check1(2 * K).Enabled = False
    End If
    K = K + 1
  End If
  If MAIN_Screen.unit_task(150).Value = True Then
    Label1(K).Caption = "T - Defend"
    Label1(K + 25).Caption = MAIN_Screen.UIC(16).Text
     If MAIN_Screen.Opt_Bnall = True Or MAIN_Screen.opt_Bnsome = True Then
      Check1(2 * K).Value = Schedule(150, 0)
      Schedule(150, 1) = 2 * K
      Schedule(150, 2) = Schedule(150, 1) + Correction
      Check1(2 * K).Enabled = True
    Else
    Schedule(150, 2) = 999
    Check1(2 * K).Enabled = False
   End If
    K = K + 1
  End If
  Counter(4) = K + 1
  Do While K < 25
  Label1(K).visible = False
  Label1(K + 25).visible = False
  Check1(2 * K).visible = False
  Check1(2 * K + 1).visible = False
  K = K + 1
  Loop
  End Sub

  Attribute VB_Name = "Resource"
  Attribute VB_Creatable = False
  Attribute VB_Exposed = False
  Option Explicit

  Private Sub Command1_Click()
  Site_Save = Table1.Text
  dtaposts.Recordset.MoveFirst
  Do While dtaposts.Recordset.Fields(1) <> Site_Save

  dtaposts.Recordset.MoveNext
  If dtaposts.Recordset.EOF = True Then
    dtaposts.Recordset.MovePrevious
    GoTo done
  End If
  Loop
  done:
```

```
M1 = dtaposts.Recordset.Fields(2)
M2 = dtaposts.Recordset.Fields(3)
Hv = dtaposts.Recordset.Fields(4)
Dims = dtaposts.Recordset.Fields(5)
Apc = dtaposts.Recordset.Fields(6)
Safs = dtaposts.Recordset.Fields(7)
Ocs = dtaposts.Recordset.Fields(8)
MAIN_Screen.Show
Unload Me

End Sub

Public Sub Form_Load()
Dim noloop As Boolean
Dim msg, title, style, response As String
noloop = True
' MAIN_Screen.Visible = False
WrongSiteTable = True
On Error GoTo Error_Mark2
If Dir(Site_File) <> "" And Site_File <> "" Then
dtaposts.DatabaseName = Site_File
dtaposts.RecordSource = "Sites"
dtaposts.Refresh
dtaposts.Recordset.MoveFirst
Resource.Show
GoTo Bottom2
End If
Error_Mark2:
msg = "Sorry - That is the Wrong Database as specified in the .ini file
- No Site Data " + Chr(13) + Chr(13) + "Please find the correct file in
the next screen."
' Style = vbOKOnly ' Define buttons.
title = "Database Error"

response = MsgBox(msg, style, title)
Resource.visible = False
    Load Tng_Site_Selection
    Tng_Site_Selection.visible = True
    Tng_Site_Selection.Show
    GoTo done:
Bottom2:

Do While dtaposts.Recordset.Fields(1) <> Site_Save

dtaposts.Recordset.MoveNext
If dtaposts.Recordset.EOF = True Then
  dtaposts.Recordset.MovePrevious
  Site_Save = dtaposts.Recordset.Fields(1)
  Table1.Text = Site_Save
  Table1.Refresh
'  GoTo done
End If
Loop

M1 = dtaposts.Recordset.Fields(2)
Text2(0).Text = M1
M2 = dtaposts.Recordset.Fields(3)
Text2(1).Text = M2
Hv = dtaposts.Recordset.Fields(4)
Text2(2).Text = Hv
Dims = dtaposts.Recordset.Fields(5)
```

```
Text2(3).Text = Dims
Apc = dtaposts.Recordset.Fields(6)
Text2(4).Text = Apc
Safs = dtaposts.Recordset.Fields(7)
Text2(6).Text = Safs
Ocs = dtaposts.Recordset.Fields(8)
Text2(5).Text = Ocs
done:
End Sub

Private Sub mnu_Return_Click()
Site_Save = Table1.Text
dtaposts.Recordset.MoveFirst
Do While dtaposts.Recordset.Fields(1) <> Site_Save

dtaposts.Recordset.MoveNext
If dtaposts.Recordset.EOF = True Then
  dtaposts.Recordset.MovePrevious
  GoTo done
End If
Loop
done:
M1 = dtaposts.Recordset.Fields(2)
M2 = dtaposts.Recordset.Fields(3)
Hv = dtaposts.Recordset.Fields(4)
Dims = dtaposts.Recordset.Fields(5)
Apc = dtaposts.Recordset.Fields(6)
Safs = dtaposts.Recordset.Fields(7)
Ocs = dtaposts.Recordset.Fields(8)
MAIN_Screen.Show
Unload Me

End Sub

Private Sub res_Data_Click()
Load Tng_Site_Selection
Tng_Site_Selection.Show
End Sub

Public Sub Update_Data()
dtaposts.Recordset.MoveFirst
Do While dtaposts.Recordset.Fields(1) <> Site_Save '
dtaposts.Recordset.MoveNext
If dtaposts.Recordset.EOF = True Then
  dtaposts.Recordset.MovePrevious
  GoTo done
End If
Loop
done:
End Sub

Private Sub Table1_Change()
Site_Save = Table1.Text
dtaposts.Recordset.MoveFirst
Do While dtaposts.Recordset.Fields(1) <> Site_Save

dtaposts.Recordset.MoveNext
If dtaposts.Recordset.EOF = True Then
  dtaposts.Recordset.MovePrevious
  Table1.Refresh
  GoTo done
```

```
End If
Loop
done:

M1 = dtaposts.Recordset.Fields(2)
M2 = dtaposts.Recordset.Fields(3)
Hv = dtaposts.Recordset.Fields(4)
Dims = dtaposts.Recordset.Fields(5)
Apc = dtaposts.Recordset.Fields(6)
Safs = dtaposts.Recordset.Fields(7)
Ocs = dtaposts.Recordset.Fields(8)

Text2(0) = M1
Text2(1) = M2
Text2(2) = Hv
Text2(3) = Dims
Text2(4) = Apc
Text2(5) = Ocs
Text2(6) = Safs
End Sub

Private Sub Table1_Click()
Site_Save = Table1.Text
dtaposts.Recordset.MoveFirst
Do While dtaposts.Recordset.Fields(1) <> Site_Save

dtaposts.Recordset.MoveNext
If dtaposts.Recordset.EOF = True Then
  dtaposts.Recordset.MovePrevious
  GoTo done
End If
Loop
done:

M1 = dtaposts.Recordset.Fields(2)
M2 = dtaposts.Recordset.Fields(3)
Hv = dtaposts.Recordset.Fields(4)
Dims = dtaposts.Recordset.Fields(5)
Apc = dtaposts.Recordset.Fields(6)
Safs = dtaposts.Recordset.Fields(7)
Ocs = dtaposts.Recordset.Fields(8)

Text2(0) = M1
Text2(1) = M2
Text2(2) = Hv
Text2(3) = Dims
Text2(4) = Apc
Text2(5) = Ocs
Text2(6) = Safs
End Sub

Attribute VB_Name = "Site_data"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Form_Load()
```

```
Text1.Text = Site_Save
Text2(0) = M1
Text2(1) = M2
Text2(2) = Dims
Text2(3) = Hv
Text2(4) = Apc
Text2(5) = Ocs
Text2(6) = Safs
End Sub

Attribute VB_Name = "Tng_Site_Selection"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Public Data_Drive, Data_Dir, Data_File, SiteFile As String
Dim I, J, K, L, M, N As Integer
Public Rating As String
Dim Resultvalue As Long
Dim Flag As Boolean
Private Sub data_Exit_Click()
If Flag = False Then
End
Else
Unload Me
End If
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
File1.Pattern = "*.mdb"
End Sub

Private Sub Drive_Cmd_Click()
File1.Pattern = "*.mdb"
ChDrive Drive1.Drive
ChDir Dir1.Path
Data_Drive = Drive1.Drive
Data_Dir = Dir1.Path
Site_File = Dir1.Path + "\" + Data_File
Tredsname.Text = Site_File
WrongSiteTable = True
Resource.dtaposts.DatabaseName = Site_File
Resource.dtaposts.RecordSource = "Sites"
Tredsname.Text = Site_File
On Error GoTo Errorhandler
Resource.dtaposts.Refresh
Resource.dtaposts.Recordset.MoveFirst
Do While Resource.dtaposts.Recordset.Fields(1) <> Site_Save

Resource.dtaposts.Recordset.MoveNext
If Resource.dtaposts.Recordset.EOF = True Then
  Resource.dtaposts.Recordset.MovePrevious
 Site_Save = Resource.dtaposts.Recordset.Fields(1)
'  Resource.Table1.Text = Site_Save
  Resource.Table1.Refresh
'  GoTo done
End If
Loop

M1 = Resource.dtaposts.Recordset.Fields(2)
Resource.Text2(0).Text = M1
M2 = Resource.dtaposts.Recordset.Fields(3)
```

```
Resource.Text2(1).Text = M2
Hv = Resource.dtaposts.Recordset.Fields(4)
Resource.Text2(2).Text = Hv
Dims = Resource.dtaposts.Recordset.Fields(5)
Resource.Text2(3).Text = Dims
Apc = Resource.dtaposts.Recordset.Fields(6)
Resource.Text2(4).Text = Apc
Safs = Resource.dtaposts.Recordset.Fields(7)
Resource.Text2(6).Text = Safs
Ocs = Resource.dtaposts.Recordset.Fields(8)
Resource.Text2(5).Text = Ocs

Unload Me
Resource.Show
GoTo Bottom

Errorhandler:
    Flag = False
msg = "Sorry - That is the Wrong Database"
Style = vbOKOnly ' Define buttons.
Title = "Database Error"

response = MsgBox(msg, Style, Title)

    MsgBox response
    WrongTable = False
    Tng_Site_Selection.Show
    Tng_Site_Selection.Database.visible = True
    Tng_Site_Selection.File1.visible = True
    Tng_Site_Selection.Tredsname.Text = Site_File
Bottom:

End Sub

Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub

Private Sub File1_Click()
Data_File = File1.filename
End Sub

Private Sub File1_DblClick()
File1.Pattern = "*.mdb"
ChDrive Drive1.Drive
ChDir Dir1.Path
Data_Drive = Drive1.Drive
Data_Dir = Dir1.Path
Site_File = Dir1.Path + "\" + Data_File
Tredsname.Text = Site_File
WrongSiteTable = True
Resource.dtaposts.DatabaseName = Site_File
Resource.dtaposts.RecordSource = "Sites"
Tredsname.Text = Site_File
On Error GoTo Errorhandler
Resource.dtaposts.Refresh
Resource.dtaposts.Recordset.MoveFirst
Do While Resource.dtaposts.Recordset.Fields(1) <> Site_Save

Resource.dtaposts.Recordset.MoveNext
If Resource.dtaposts.Recordset.EOF = True Then
```

```
  Resource.dtaposts.Recordset.MovePrevious
 Site_Save = Resource.dtaposts.Recordset.Fields(1)
'  Resource.Table1.Text = Site_Save
  Resource.Table1.Refresh
'  GoTo done
End If
Loop

M1 = Resource.dtaposts.Recordset.Fields(2)
Resource.Text2(0).Text = M1
M2 = Resource.dtaposts.Recordset.Fields(3)
Resource.Text2(1).Text = M2
Hv = Resource.dtaposts.Recordset.Fields(4)
Resource.Text2(2).Text = Hv
Dims = Resource.dtaposts.Recordset.Fields(5)
Resource.Text2(3).Text = Dims
Apc = Resource.dtaposts.Recordset.Fields(6)
Resource.Text2(4).Text = Apc
Safs = Resource.dtaposts.Recordset.Fields(7)
Resource.Text2(6).Text = Safs
Ocs = Resource.dtaposts.Recordset.Fields(8)
Resource.Text2(5).Text = Ocs

Unload Me
Resource.Show
GoTo Bottom

Errorhandler:
    Flag = False
msg = "Sorry - That is the Wrong Database"
Style = vbOKOnly ' Define buttons.
Title = "Database Error"

response = MsgBox(msg, Style, Title)

    MsgBox response
    WrongTable = False
    Tng_Site_Selection.Show
    Tng_Site_Selection.Database.visible = True
    Tng_Site_Selection.File1.visible = True
    Tng_Site_Selection.Tredsname.Text = Site_File
Bottom:

End Sub

Private Sub Form_Load()
ChDrive Drive1.Drive
ChDir Dir1.Path
File1.Pattern = "*.mdb"
Flag = True
WrongSiteTable = True
End Sub
```